TESIS DE CARRERA DE MAESTRÍA EN CIENCIAS FÍSICA

OPTIMIZACIÓN DE BASES STURMIANAS PARA EL PROBLEMA DE TRES CUERPOS CUÁNTICO.

Lic. Federico Turco
Maestrando

Dr. Flavio D. Colavecchia
Director

Miembros del Jurado
Dra. Renata Della Picca (Instituto Balseiro, Centro Atómico Bariloche)
Dr. Alejandro Kolton (Instituto Balseiro, Centro Atómico Bariloche)
Dr. Juan Martín Randazzo (Instituto Balseiro, Centro Atómico Bariloche)

Marzo de 2017

División Física Atómica, Molecular y Óptica
Centro Atómico Bariloche

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

Índice de contenidos

Índice de figuras

Índice de cuadros

Resumen

La importancia del problema de tres cuerpos cuánticos en la física de colisiones atómicas es bien conocida. Una de las maneras de resolverlo es a travéz del método CI (Configuración Interacción). En términos generales, este método utiliza una base del problema de dos cuerpos para buscar una solución al problema de tres cuerpos como combinación lineal de los elementos de la misma. El objetivo de esta tesis es el desarrollo y la optimización de códigos numéricos para el cálculo eficiente de las bases de funciones Sturmianas, que son soluciones de una ecuación de Schrödinger modelo, donde se asume que la energía es fija, y tomando como autovalor a la magnitud de la interacción. Dichas bases se pueden obtener numéricamente como solución de un problema de autovalores y autovectores de una matriz tridiagonal, simétrica y con un único elemento complejo en su diagonal impuesto por la condición de borde. Los autovalores se obtienen mediante el método de iteración QR mientras que, utilizando el método de la potencia inversa, se calculan los autovectores asociados. El cálculo de autovalores se implementó en lenguaje C para su funcionamiento en la CPU, y el cálculo de autovectores se implementó tanto para CPU, como para funcionar completamente en procesadores gráficos (GPU) utilizando la arquitectura CUDA. Se utilizaron distintas características de esta biblioteca que permiten la ejecución asincrónica de códigos en GPU y CPU. Cada vez que el algoritmo QR obtiene un autovalor, se obtiene el autovector en GPU. Se corroboró el correcto funcionamiento del cálculo de las bases Sturmianas y se obtuvo un resultado favorable en la eficiencia del híbrido CPU-GPU respecto al cálculo sólo en la CPU. En efecto, el tiempo empleado para el cálculo de los autovalores y autovectores en el código híbrido CPU-GPU no difiere a los fines prácticos del necesario para calcular sólo los autovalores en la CPU.

Abstract

The three body quantum problem is very important in collisions physics. A common way of solving it is to use the configuration interaction method (CI), in general terms this method uses a basis from the two bodies problem to expand the original one with linear combination of the basis elements. The main goal of this work is the development and optimization of the numerical codes used for solving the Schrödinger equation for two bodies. The basis obtained is called *Sturmian* and is formed by the set of solutions where the energy is fixed and the eigenvalue is taken as the magnitude of interaction. The basis can be found by solving numerically a problem of eigenvalues and eigenvectors of a symmetric and tridiagonal matrix with just one complex value in the last diagonal element due to border condition. The eigenvalues are obtained by using the QR method while the eigenvectors are computed by using the inverse power method. The algorithm to find eigenvalues was implemented in the C programming language to work in the CPU. At the same time, the eigenvectors algorithm can work in CPU or in the graphic processor unit using CUDA. Different characteristics of CUDA are used to allow the asynchronous call of the functions which work on the GPU. This mean that each time the program locates some eigenvalue, then a calculation is asynchronously launched in the GPU to find the corresponding eigenvector. It was checked the correct work by calculating different Sturmians basis for well-known problems and comparing with the literature. Besides there was a good improvement on the efficiency of the CPU-GPU hybrid code respect to just calculate the basis in the CPU. In fact, the time used to find basis in the GPU does not differ from the ones used for eigenvalues in the CPU.

Capítulo 1

Introducción

El problema de tres cuerpos cuántico tiene gran relevancia desde el punto de vista de las colisiones atómicas. Por ejemplo, los estados ligados del átomo de helio, los procesos de doble fotoionización o ionización por colisión de iones de un blanco atómico pueden modelarse como un problema de tres cuerpos cuántico. La principal dificultad del mismo radica en que no puede resolverse exactamente de manera analítica, salvo en algunos pocos casos sencillos.

Existe una gran cantidad de métodos para resolver de forma aproximada el problema de tres cuerpos. Entre los diferentes algoritmos se puede mencionar el método variacional, que utiliza las autofunciones del átomo de hidrógeno como base y diferentes parámetros se busca minimizar la energía, obteniendo así los estados ligados del sistema. Por otro lado se encuentra el cálculo numérico, de gran interés debido a la posibilidad de resolver el problema utilizando un potencial arbitrario, para los cuales no se conoce solución analítica. También debido a la opción de imponer las condiciones de borde necesarias, sin tener que desarrollar nuevamente una teoría consistente.

Para el problema de tres cuerpos se utiliza una técnica llamada *Configuración Interacción*[1], la cual utiliza resultados del problema de dos cuerpos para desarrollar la solución en cada punto del espacio. Más detalles de este método se presentan en la Sección 1.1.

En este trabajo se resuelve la ecuación de Schrödinger para el caso particular de dos cuerpos que interactuan mediante un potencial central. Dicho potencial se considera como una combinación de uno de largo alcance (potencial auxiliar) y otro de corto alcance (potencial generador), suponiendo la energía del sistema bien definida y de magnitud E. Dejando variable un factor multiplicativo al potencial generador, se buscan los autovalores que producen que el sistema posea energía E y sus estados (funciones de onda) asociados. El conjunto de dichos estados se llama *Base Sturmiana* por su conección con el problema de Sturm-Liouville.

En el capítulo 1, se presenta una introducción teórica al problema planteado y a

las bases Sturmianas. En el capítulo 2 se explican los algoritmos utilizados para la obtención de los autovalores y autovectores que forman la base Sturmiana, tanto en la CPU como en la GPU. En el capítulo 3 se muestran los resultados obtenidos, analizando precisión y eficiencia temporal.

En toda la tesis se utilizan unidades atómicas (u.a.) para simplificar la presentación de las ecuaciones, un resumen de las unidades más utilizadas se encuentra en la siguiente tabla [2]:

Unidades Atómicas	
longitud	a_B =5.29x10^{-11} m
velocidad	v_B =2.188x10^{6} m/s
tiempo	2.42x10^{-17} s
masa	m_e =9,109x10^{-31} kg
$\hbar$	1 u.a.

Tabla 1.1: Unidades Atómicas

1.1. Problema de tres cuerpos cuánticos

En esta sección se presenta de forma breve y resumida un método de resolución del problema de tres cuerpos, mostrando así la necesidad de optimizar el cálculo de las bases Sturmianas de dos cuerpos. Los detalles de las ecuaciones presentadas a continuación pueden encontrarse en [3].

Se considera un sistema de tres cuerpos, de masa m_1, m_2 y m_3, para la resolución se utiliza un sistema de *coordenadas de Jacobi* $\{r_{i,j},R_{k,ij}\}$ (Ver Figura 1.1), reescalado por las masas de la forma:

$$\begin{aligned} x_k &= \left(\frac{\mu_{i,j}}{\mu}\right)^{\frac{1}{2}} r_{i,j} \\ X_k &= \left(\frac{\mu}{\mu_{i,j}}\right)^{\frac{1}{2}} R_{k,ij} \end{aligned} \tag{1.1}$$

con $\mu_{i,j} = \frac{m_i\, m_j}{m_i+m_j}$ y $\mu = \frac{m_i\, m_j\, m_k}{(m_i+m_j+m_k)^{1/2}}$. Utilizando dicho cambio de variables se pueden definir las denominadas *coordenadas hiperesféricas*: el *hiper-radio* (ρ) y el *hiper-ángulo* (α) dados por:

$$\begin{aligned} \rho^2 &= x_k^2 + X_k^2 \\ \tan\alpha &= \frac{X_k}{x_k} \end{aligned} \tag{1.2}$$

Son necesarias además, otras cuatro coordenadas Θ_{X_k},ϕ_{X_k} y Θ_{x_k},ϕ_{x_k}, que representan los ángulos que forman X_k y x_k con el centro de masa, respectivamente.

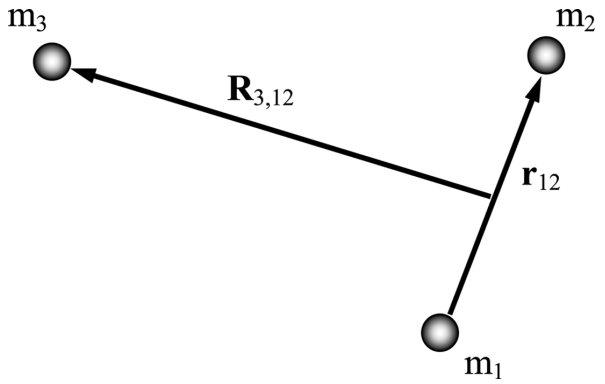

Figura 1.1: Coordenadas de Jacobi para tres cuerpos.

La ecuación de Schrödinger para tres cuerpos cuánticos está dada por:

$$(T + V - E)\,\Psi(\rho, \alpha, \widehat{\mathbf{X}}, \widehat{\mathbf{x}}) = 0 \tag{1.3}$$

donde E es la energía del sistema, V es el potencial que media la interacción entre las partículas y por simplicidad se utiliza $\widehat{\mathbf{X}}$ y $\widehat{\mathbf{x}}$ para representar $\{\Theta_{X_k}, \phi_{X_k}\}$ y $\{\Theta_{x_k}, \phi_{x_k}\}$ respectivamente. Por otro lado, T es el operador de energía cinética definido en dichas coordenadas como:

$$T = -\frac{1}{2\,\mu}\left[\frac{1}{\rho^5}\frac{\partial}{\partial\rho}\left(\rho^5\frac{\partial}{\partial\rho}\right) - \frac{\Lambda^2}{\rho^2}\right]. \tag{1.4}$$

y el operador Λ^2 es el *gran momento angular orbital*, dado por:

$$\Lambda^2 = -\frac{1}{\sin^2\alpha\,\cos^2\alpha}\frac{d}{d\alpha}\left(\sin^2\alpha\,\cos^2\alpha\,\frac{d}{d\alpha}\right) + \frac{\mathbf{j}^2}{\cos^2\alpha} + \frac{\mathbf{l}^2}{\sin^2\alpha} \tag{1.5}$$

con $\mathbf{j}$ y $\mathbf{l}$ los momentos angulares rotacional y centrífugo.

Por otro lado, a la iteracción entre la particula se la define como una suma de dos potenciales de la forma:

$$V = \mathcal{V}(\rho, \alpha) + U(\rho), \tag{1.6}$$

donde para fines prácticos se considera al potencial U como función únicamente del hiper-radio. Además se pide que estos nuevos potenciales sean combinación de un potencial de largo y otro de corto alcance, de la forma:

$$\begin{aligned} \mathcal{V}(\rho, \alpha) &= \mathcal{V}_1(\rho, \alpha) + \mathcal{V}_2(\rho, \alpha) \\ U(\rho) &= U_1(\rho) + \beta U_2(\rho) \end{aligned} \tag{1.7}$$

donde se incluye el parámetro β por conveniencia para los resultados que se obtienen más adelante. Los potenciales $U_2(\rho)$ y $\mathcal{V}_2(\rho,\alpha)$ son de corto alcance y $U_1(\rho)$ y $\mathcal{V}_1(\rho,\alpha)$ son de largo alcance, asumiento que tienen forma Coulombiana en el hiper-radio, dada por:

$$\begin{aligned} U_1(\rho) &= U_0(\rho) - \frac{Z}{\rho} \\ \mathcal{V}_1(\rho,\alpha) &= -\frac{C(\alpha)}{\rho} \end{aligned} \tag{1.8}$$

con $C(\alpha)$ una función del hiper-ángulo y U_0 de corto alcance. Reemplazando dichos potenciales en la Ecuación (1.3) se obtiene una ecuación diferencial que puede ser resuelta utilizando variables separadas, se propone una solución general dada por la expresión:

$$\Psi(\rho,\alpha,\widehat{\mathbf{X}},\widehat{\mathbf{x}}) = \mathcal{R}(\rho)\,\Gamma(\alpha)\,\mathcal{X}(\widehat{\mathbf{X}})\,\chi(\widehat{\mathbf{x}}) \tag{1.9}$$

donde se obtiene la solución exacta para las funciones angulares mientras que la dependencia en las variables hiperesféricas se obtiene de la forma:

$$\begin{aligned} \mathcal{R}(\rho) &= \frac{S_{\beta,\lambda}(\rho)}{\rho^{5/2}} \\ \Gamma(\alpha) &= \frac{H_{\lambda,j,l}(\alpha)}{\sin\alpha\,\cos\alpha} \\ \mathcal{X}(\widehat{\mathbf{X}}) &= Y_l^{m_l}(\widehat{\mathbf{X}}) \\ \chi(\widehat{\mathbf{x}}) &= Y_j^{m_j}(\widehat{\mathbf{x}}) \end{aligned} \tag{1.10}$$

En la expresión anterior se identifican los *Armónicos Esféricos* y además dos funciones ($S_{\beta,\lambda}$ y $H_{\lambda,j,l}$) que están determinadas según el conjunto de ecuaciones dadas por:

$$\begin{aligned} \left[-\frac{d^2}{d\alpha_k^2} + \frac{j\,(j+1)}{\cos^2\alpha} + \frac{l\,(l+1)}{\sin^2\alpha} + 2\mu\,r_c\,C(\alpha) + 2\mu\,r_\nu^2\mathcal{V}_2(r_\nu,\alpha)\right] H_{\lambda,j,l}(\alpha) \\ = (\lambda+2)^2\,H_{\lambda,j,l}(\alpha) \end{aligned} \tag{1.11}$$

$$\left[-\frac{1}{2\,\mu}\frac{d^2}{d\rho^2} + \frac{\lambda\,(\lambda+4) - \frac{15}{4}}{2\mu\rho^2} + U_1(\rho) - E\right] S_{\beta,\lambda}(\rho) = \beta\,U_2(\rho)\,S_{\beta,\lambda}(\rho) \tag{1.12}$$

donde r_c y r_ν son parámetros que se asumen constante durante el cálculo. La constante β se toma como autovalor de la ecuación diferencial en la expresión (1.12), así como el

factor λ en la ecuación (1.11).

Las condiciones de borde de la función $H_{\lambda,j,l}$ estan definidas por la necesidad de regularidad de las soluciones dadas en la expresión (1.10), dado que $\frac{H_{\lambda,j,l}(\alpha)}{\sin\alpha\,\cos\alpha}$ no debe diverger en $\alpha = 0$ y $\alpha = \pi/2$ se tienen las siguientes condiciones:

$$H_{\lambda,j,l}(0) = H_{\lambda,j,l}\left(\frac{\pi}{2}\right) = 0 \tag{1.13}$$

Por otro lado la función que posee dependencia en el hiper-radio $\left(\frac{S_{\beta,\lambda}(\rho)}{\rho^{5/2}}\right)$ no puede diverger en el origen, entonces se debe cumplir la condición $S_{\beta,\lambda}(\rho) = \rho^l$ con $l > 5/2$. Además se toma como condición de borde que para radios suficientemente grandes se debe cumplir la ecuación (1.12) considerando que los potenciales de corto alcance (U_0 y U_2) se anulan. Las soluciones para dicha ecuación varían según el signo de la energía, dichos casos se presentan en la Sección (1.2.2).

Como se verá más adelante (Sección 1.2, Ecuacion 1.24 y 1.25) las soluciones obtenidas para $H_{\lambda,j,l}$ y $S_{\beta,\lambda}$ forman una base del espacio L^2, por lo tanto la solución al problema de tres cuerpos se puede escribir como combinación lineal de los elementos de dicha base, de forma general esto es:

$$\Psi(\rho,\alpha,\widehat{\mathbf{X}},\widehat{\mathbf{x}}) = \sum_{\beta_i,\lambda_n,j,m_j,l,m_l} a_{\beta_i,\lambda_n,j,m_j,l,m_l}\,\frac{S_{\beta_i,\lambda_n}(\rho)}{\rho^{5/2}}\,\frac{H_{\lambda_n,j,l}(\alpha)}{\sin\alpha\cos\alpha}\,Y_j^{m_j}(\widehat{\mathbf{x}})\,Y_l^{m_l}(\widehat{\mathbf{X}}) \tag{1.14}$$

De este modo, para la resolución del problema de tres cuerpos cuánticos es necesario conocer las soluciones correspondientes para $H_{\lambda,j,l}(\alpha)$ y $S_{\beta,\lambda}(\rho)$. Como se muestra más adelante (Sección 1.3), las ecuaciones (1.11) y (1.12) tienen la misma forma que la ecuación de Schrödinger para la dependencia radial del problema de dos cuerpos con potencial central. Esto motiva al desarrollo de códigos eficiente para el cálculo de las soluciones para el problema de dos cuerpos.

1.2. Bases Sturmianas en mecánica cuántica

Como se presentó en la sección anterior, uno de los métodos de resolución del problema de tres cuerpos cuánticos consiste en una expansión en funciones que se obtienen como solución de una ecuación diferencial de segundo orden, similar a la ecuación de Schrödinger de dos cuerpos. En lo que resta de este capítulo se presenta el estudio de dicha ecuación y la base obtenida con las soluciones correspondientes.

Numerosos problemas de física se resuelven buscando una base adecuada para la expansión de una determinada función; un claro ejemplo es la resolución de la ecuación

de Schrödinger para encontrar autoenergías y las bases asociadas. Esto plantea un problema ya que la expansión en dicha base consta de una suma infinita sobre los autoestados discretos y una integral sobre los estados continuos.

Una forma de encontrar una base adecuada es utilizar una energía fija y tomar como autovalor a la magnitud del potencial, siendo los autovectores asociados elementos de la misma. Este procedimiento da lugar a las denominadas *funciones Sturmianas Generalizadas*, cuyo cálculo constituye el tema de esta tesis.

Para resolver el caso de dos partículas, con interacción dependiente sólo de la distancia relativa, puede utilizarse un sistema de coordenadas solidario al centro de masa. Teniendo en cuenta que el potencial es central ($V(r)$ con $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$), la ecuación de Schrödinger para dos partículas queda expresada de la forma:

$$\left[-\frac{1}{2\,\mu}\nabla^2 + V_{Total}(\mathbf{r}) - E\right]\psi(\mathbf{r}) = 0, \tag{1.15}$$

En este trabajo se utiliza $r = |\mathbf{r}|$, o equivalentemente $\mathbf{r} = r\widehat{\mathbf{r}}$; en $\widehat{\mathbf{r}}$ se encuentra toda la dependencia angular. Se consideró a $\mathbf{r}_1$ y $\mathbf{r}_2$ a la posición de la partícula 1 y 2 respecto del centro de masa, respectivamente; $\mu = \frac{m_1\,m_2}{m_1+m_2}$ es la masa reducida del sistema y E es su energía.

Como se comentó anteriormente no se desean las energías y sus autoestados, sino que se reemplaza el potencial por uno más general, de la forma:

$$V_{Total}(\mathbf{r}) = U(\mathbf{r}) + \beta\,V(\mathbf{r}). \tag{1.16}$$

El potencial $V(r)$, denominado *potencial generador* es de corto alcance, mientras que el potencial $U(r)$ es llamado *potencial auxiliar* y se asume de largo alcance (Coulombiano). Siendo este último utilizado para imponer las condiciones de borde deseadas, usualmente asociadas a procesos de colisión que involucran partículas cargadas. La magnitud del potencial generador β pasa a ser considerada como el autovalor del problema, es decir, se buscan los distintos β que satisfacen la ecuación (1.15) manteniendo la energía E fija.

Los potenciales auxiliar y generador se elijen para satisfacer determinadas condiciones físicas relacionadas con el problema específico que se está considerando. En general, se asume que el potencial auxiliar se comporta de la forma $U(r) = U_{int}(r) + \frac{Z}{r}$, siendo $U_{int}(r)$ un potencial de alcance r_c. Por otro lado, el potencial generador debe tener el mismo alcance y nunca anularse para radios menores a r_c. Dichas condiciones se pueden expresar como

$$\begin{aligned} U_{int}(r > r_c) =& 0 \\ V(r > r_c) =& 0 \\ V(r) \neq& 0, \qquad \forall r < r_C \end{aligned} \tag{1.17}$$

Para una energía fija E, el conjunto de autovalores β_n tiene asociado un conjunto de autofunciones ψ_n, que son las funciones Sturmianas Generalizadas.

La ecuación (1.15) se resuelve mediante separación de variables, utilizando coordenadas esféricas, mediante la transformación:

$$\begin{aligned} x &= r\ \sin\,\theta\ \cos\,\phi \\ y &= r\ \sin\,\theta\ \sin\,\phi \\ z &= r\ \cos\,\theta \end{aligned} \tag{1.18}$$

Modificando la ecuación (1.15) para trabajar en coordenadas esféricas y considerando una solución del tipo:

$$\psi(\mathbf{r}) = \frac{S_{n,l}(r)}{r}\Theta_{l,m}(\theta,\phi) \tag{1.19}$$

se obtiene para la parte radial $S_{n,l}(r)$, la ecuación diferencial:

$$\left[-\frac{1}{2\,\mu}\frac{d^2}{dr^2} + \frac{l(l+1)}{2\,\mu\,r^2} + U(r) + \beta_{n,l}\,V(r) - E\right] S_{n,l}(r) = 0 \tag{1.20}$$

mientras la parte dependiente de la dirección del versor $\widehat{\mathbf{r}}$, satisface la ecuación:

$$\frac{1}{\sin\theta}\frac{\partial}{\partial\,\theta}\left(\sin\theta\frac{\partial\Theta_{l,m}(\theta,\phi)}{\partial\,\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2\,\Theta_{l,m}(\theta,\phi)}{\partial\,\phi^2} + l(l+1)\Theta_{l,m}(\theta,\phi) = 0 \tag{1.21}$$

Para la ecuación (1.21) las soluciones son conocidas y reciben el nombre de *Armónicos Esféricos*, $\Theta_{l,m}(\theta,\phi) = Y_l^m(\theta,\phi)$:

$$Y_l^m(\theta,\phi) = \sqrt{\frac{(2l+1)\,(l-m)!}{4\pi\,(l+m)!}}\,e^{im\phi}\,\frac{(-1)^m}{2^l\,l!}(1-\cos^2\theta)^{m/2}\frac{d^{l+m}}{dr^{l+m}}(\cos^2\theta-1)^l. \tag{1.22}$$

De este modo la solución a la ecuación (1.15) para determinados números cuánticos n, l, m tiene la forma $\psi_{n\,l\,m}(r,\theta,\phi) = \frac{S_{n,l}(r)}{r}\,Y_l^m(\theta,\phi)$ y una solución general está dada por:

$$\psi(\mathbf{r}) = \psi(r,\theta,\phi) \; = \sum_{n,l,m} a_{n,l,m} \, \frac{S_{n,l}(r)}{r} \, Y_l^m(\theta,\phi). \tag{1.23}$$

Las soluciones para la función de onda angular no dependen de los potenciales U y V utilizados, siempre y cuando los potenciales posean simetría esférica. Teniendo en cuenta lo anterior, la solución de la ecuación de Schrödinger se reduce a resolver la ecuación diferencial (1.20). En los Capítulos 2 y 3 se presentan los algoritmos utilizados para realizar la tarea.

Dado que la ecuación (1.20) con las condiciones de borde utilizadas es un caso particular de una ecuación de Sturm-Liouville [4], esto asegura la ortogonalidad de las autofunciones encontradas con función de peso $V(r)$, para energías negativas:

$$\begin{aligned} \int_0^{r_c} S_{n,l}(r)\, V(r)\, S_{n',l}(r)\, dx &= \delta_{n,n'} \\ \sum_n S_{n,l}(r)\, V(r')\, S_{n,l}(r') &= \delta(r-r') \end{aligned} \tag{1.24}$$

donde $\delta_{n,n'}$ representa la *delta de Kronecker* y $\delta(r-r')$ es la denominada *Delta de Dirac*. Por otra parte, para energías positivas y condiciones de borde de onda estacionarias se tiene un espectro continuo de autovalores β_n, y se tiene:

$$\begin{aligned} \int_0^{r_c} S_{\nu\, l}(r)\, V(r)\, S_{\nu',l}(r)\, dx &= \delta(\nu-\nu') \\ \int_0^{\infty} S_{\nu,l}(r)\, V(r')\, S_{\nu,l}(r')\, d\nu &= \delta(r-r') \end{aligned} \tag{1.25}$$

Además teniendo en cuenta la ortogonalidad y clausura de los Armónicos Esféricos y la representación de $\psi_{n,l,m}$ en términos de las funciones Sturmianas Generalizadas se obtiene la relación de ortogonalidad:

$$\coprod_{\nu} \psi_\nu(\mathbf{r})\, V(r)\, \psi_\nu(\mathbf{r}') = \delta(\mathbf{r}-\mathbf{r}'), \tag{1.26}$$

donde el operador $\coprod_\nu$ indica la suma sobre todos los ν para el caso de energías negativas e indica la integral sobre el espectro continuo de ν cuando es positiva.

1.2.1. Aplicaciones de las funciones Sturmianas

Entre las numerosas aplicaciones de las funciones Sturmianas Generalizadas definidas anteriormente se detallará a modo de ejemplo la obtención de los estados ligados asociados a un potencial $U(r)$ con simetría esférica.

Los estados ligados de un potencial $U(r)$ dado, son solución de la ecuación de Schrödinger:

$$\left[-\frac{1}{2\,\mu}\nabla^2 + U(r) - E\right]\psi(\mathbf{r}) = 0. \tag{1.27}$$

Se puede empezar la resolución del problema planteando las bases de Sturmianas encontradas utilizando la misma masa reducida, una energía fija (E_0), un potencial auxiliar $U(r)$ y un potencial generador $V(r)$. De este modo los elementos de la bases son solución de:

$$\left[-\frac{1}{2\mu}\nabla^2 + U(r) + \beta_{n,l}V(r) - E_0\right]\frac{S_{n,l}(r)}{r}Y_l^m(\widehat{\mathbf{r}}) = 0. \tag{1.28}$$

Teniendo en cuenta la semejanza entre las ecuaciones (1.27) y (1.28), se propone una solución general para la ecuación (1.27) en término de los elementos de la base Sturmiana de la forma:

$$\psi(\mathbf{r}) = \sum_{n,l,m} a_{n,l,m}\,\frac{S_{n,l}(r)}{r}Y_l^m(\widehat{\mathbf{r}}) \tag{1.29}$$

Despejando de la ecuación (1.28), se obtiene que las funciones Sturmianas satisfacen:

$$\left(-\frac{1}{2\mu}\nabla^2 + U(r)\right)\frac{S_{n,l}(r)}{r}Y_l^m(\widehat{\mathbf{r}}) = \left(-\beta_{n,l}V(r) + E_0\right)\frac{S_{n,l}(r)}{r}Y_l^m(\widehat{\mathbf{r}}) \tag{1.30}$$

y reemplazando en la ecuación (1.27) la solución general definida por (1.29) y utilizando (1.30) se obtiene la ecuación:

$$\sum_{n,l,m} a_{n,l,m}\left(E_0 - \beta_{n,l}\,V(r)\right)\frac{S_{n,l}(r)}{r}Y_l^m(\widehat{\mathbf{r}}) = \sum_{n,l,m} a_{n,l,m}\,E\,\frac{S_{n,l}(r)}{r}Y_l^m(\widehat{\mathbf{r}}) \tag{1.31}$$

luego proyectando (1.31) por izquierda con $\frac{S_{n',l'}(r)}{r}Y_{l'}^{m'*}(\widehat{\mathbf{r}})$ y utilizando la propiedad de ortogonalidad de los Armónicos Esféricos se obtiene:

$$\begin{aligned}\sum_{n,l,m} a_{n,l,m}\int_0^\infty S_{n,l}(r)(E_0 - \beta_{n,l}V(r))\,S_{n',l'}(r)\,dr\delta_{l,l'}\delta_{m,m'} = \\ = \sum_{n,l,m} a_{n,l,m}\,E\int_0^\infty S_{n,l}(r)S_{n',l'}(r)\,dr\,\delta_{l,l'}\delta_{m,m'}.\end{aligned} \tag{1.32}$$

Esto determina un sistema de N ecuaciones para cada par l', m', donde N es el tamaño de la base utilizada. Se puede escribir de forma matricial si se considera que

las integrales definen elementos de matriz:

$$\begin{aligned}[\mathbf{S}]_{n,n'} &= \int_0^\infty S_{n,l}(r)\, S_{n',l}(r)\, dr \\ [\mathbf{V}]_{n,n'} &= -\int_0^\infty S_{n,l}(r)\beta_{n,l}V(r)S_{n',l}(r)\, dr = -\beta_{n,l}\delta_{n,n'}.\end{aligned} \tag{1.33}$$

La representación matricial del potencial es diagonal por las propiedades de ortogonalidad de las bases Sturmianas. Sin embargo la matriz de solapamiento ($[\mathbf{S}]$) es simétrica pero no será, en principio, diagonal. Está demostrado que la integral $[\mathbf{S}]_{n,n'}$ converge cualquiera sea la energía y las condiciones de borde [5, pág 4].

Reemplazando (1.33) en (1.32), se obtiene una ecuación matricial de la forma:

$$(\mathbf{V}\,\mathbf{S}^{-1} + E_0\,\mathbf{I})\mathbf{a}'_{l,m} = E\,\mathbf{a}'_{l,m} \tag{1.34}$$

con $\mathbf{a}'_{l,m} = (a_{1,l,m}, a_{2,l,m}, \cdots, a_{N,l,m})^T$.

De este modo se determinan los coeficientes de la expansión (1.29) resolviendo los autovectores correspondiente de la ecuación (1.34). Y se obtienen las autoenergías E del sistema directamente como los autovalores del problema 1.34.

1.2.2. Límites asintóticos para el potencial Coulombiano

Entre los diferentes potenciales auxiliares que pueden ser utilizados para generar una base Sturmiana se encuentra el potencial coulombiano, las funciones generadas de este modo poseen el nombre de *funciones Sturmianas Coulombianas* (FSC). Dado que el potencial generador es de corto alcance, la solución de la ecuación 1.20, se debe comportar como la solución teórica al potencial coulombiano para radios suficientemente grandes. Dado que la ecuación de Sturm-Liouville precisa de dos condiciones de borde, se utiliza el límite asintótico correspondiente.

Partiendo de la ecuación (1.20) y considerando que los potenciales de corto alcance se anulan ($U_{int}(r > r_c) + \beta\, V(r > r_c) = 0$) se obtiene que para grandes distancias se satisface:

$$\left[-\frac{1}{2\,\mu}\frac{d^2}{dr^2} + \frac{l(l+1)}{2\,\mu\,r^2} + \frac{Z}{r} - E\right] S(r) = 0. \tag{1.35}$$

La solución analítica de dicha ecuación es conocida y en el límite asintótico está dada por [6, pág 139]:

$$\lim_{r\to\infty} S^{\pm}(r) = e^{\pm\, i(kr - \frac{Z\,\mu}{k} ln(2\,k\,r))} \tag{1.36}$$

donde $k = \sqrt{2\,E\,\mu}$.

Si el sistema tiene energía negativa, el límite asintótico $S^{\pm}$ tiene una única solución posible para evitar la divergencia de la función de onda, y es [4]:

$$\lim_{r\to\infty} S^{+}(r) = e^{-\kappa r - \frac{Z\,\mu}{\kappa} ln(2\kappa r)} \tag{1.37}$$

con $\kappa = \sqrt{2\,E\,\mu}$. Esta solución corresponde a autoestados estacionarios, ya que el flujo radial de la función de onda se anula (por ser puramente real, $j_r^{\psi} = 0$). Mientras para energías positivas el sistema tiene ambas soluciones dadas por la ecuación (1.36) que representan autoestados que poseen flujo de partículas entrante o saliente según el signo negativo y positivo, respectivamente. El flujo asociado a la función de onda está dado por $j_r^{\psi} = \pm\,\frac{1}{r^2}(k - \frac{Z\,\mu}{kr})$, por lo que integrando en una superficie esférica de radio r, se observa un flujo neto $2k$.

Dado que toda las soluciones asintóticas tienen el mismo comportamiento se espera que la solución al sistema descrito como combinación de dichas funciones posea también el mismo comportamiento, ya que la única dependencia está en la energía.s

1.3. Relación con el problema de tres cuerpos

Para concluir el la introducción a las bases Sturmianas se mostrará la relación entre el problema de dos cuerpos presentado en las sección anterior y el problema de tres cuerpos, teniendo en cuenta que las soluciones $H_{\lambda,j,l}(\alpha)$ y $S_{\beta,\lambda}(\rho)$ definidas por las ecuaciones 1.11 y 1.12.

Comparando las ecuaciones nombradas anteriormente con la ecuación de schrödinger radial de dos cuerpos (ecuacion 1.20) se observa que las mismas son equivalentes utilizando 1.11 y considerando:

$$\begin{aligned} l\,(l+1) &= \lambda\,(\lambda+1) - \frac{15}{4} \\ V(r) &= U_2(r) \\ U(r) &= U_1(r), \end{aligned} \tag{1.38}$$

mientras que para la ecuación 1.12 se puede utilizar una relación similar, siendo el sistema el mismo que considerando (en 1.20) momento angular nulo (l=0), potencial generador constante y potencial auxiliar dado por:

$$U(r) = \frac{j\,(j+1)}{\cos^2\alpha} + \frac{l\,(l+1)}{\sin^2\alpha} + 2\mu\,r_c\,C(\alpha) + 2\mu\,r_\nu^2\mathcal{V}_2(r_\nu,\alpha). \tag{1.39}$$

De este modo dichas ecuaciones son equivalentes a 1.20, por los tanto los algoritmos utilizados para resolverlas son los mismos que se desarrollarán en el siguiente capítulo.

Capítulo 2

Implementación Numérica

La ecuación de Schrödinger radial (1.20) que define la solución al problema de dos cuerpos cuánticos puede resolverse de diversas maneras, pero el número de potenciales para los cuales se pueden encontrar soluciones analíticas es reducido. Para ello se proponen soluciones numéricas en el marco del método de diferencias finitas, esto es, discretizando la coordenada radial y reescribiendo la ecuación con aproximaciones de la derivada segunda como se explicará a continuación.

2.1. Discretización espacial

A fin de resolver la ecuación (1.20) usando diferencias finitas, se implementa una discretización del espacio en N intervalos iguales de longitud dr, entre valores iniciales $r_0 = 0$ y $r_N = N\,dr$, siendo r_N el radio máximo donde consideraremos al potencial generador suficientemente débil.

Se calcula numéricamente una aproximación de la derivada segunda de una función arbitraria (Φ) en la grilla utilizando diferencias finitas, partiendo de la ecuación:

$$\begin{aligned}\Phi_{i+1} &= \Phi_i + \frac{d\,\Phi_i}{dr}\,dr + \frac{1}{2}\frac{d^2\,\Phi_i}{dr^2}\,dr^2 + \frac{1}{6}\frac{d^3\,\Phi_i}{dr^3}\,dr^3 + \mathcal{O}(dr^4)\\ \Phi_{i-1} &= \Phi_i - \frac{d\,\Phi_i}{dr}\,dr + \frac{1}{2}\frac{d^2\,\Phi_i}{dr^2}\,dr^2 - \frac{1}{6}\frac{d^3\,\Phi_i}{dr^3}\,dr^3 + \mathcal{O}(dr^4)\end{aligned} \tag{2.1}$$

donde el subíndice $i = 1, 2, 3, ..., \infty$ indica la evaluación de la función Φ en la posición $r_i = i\,dr$.

Realizando la suma entre las ecuaciones anteriores se obtiene una aproximación de segundo orden para la derivada segunda de Φ en la posición r_i, siendo el resultado:

$$\frac{d^2\,\Phi_i}{dr^2} = \frac{\Phi_{i+1} - 2\Phi_i + \Phi_{i-1}}{dr^2} + \mathcal{O}(dr^2). \tag{2.2}$$

Reemplazando en la ecuación de Schrödinger radial y utilizando $\Phi_i = S_{n,l}(r_i) = S_i$ (donde obviaremos los números cuánticos l y n) se obtiene una expresión para la ecuación radial (1.20) dada por un sistema de ecuaciones lineales en cada punto de la grilla:

$$-\frac{1}{2\,\mu}\,\frac{S_{i+1}-2\,S_i+S_{i-1}}{dr^2}+\frac{l(l+1)\,S_i}{2\,\mu\,{r_i}^2}+U_i\,S_i-E\,S_i=-\beta\,V_i\,S_i \tag{2.3}$$

donde $r_i = i\,dr$, $S_i \equiv S_{n,l}(r_i)$, $V_i \equiv V(r_i)$ y $U_i \equiv U(r_i)$ son la función de onda, potencial generador y el potencial auxiliar, respectivamente.

La expresión (2.3) es una ecuación recurrente donde la solución en cada punto de la grilla depende directamente de los adyacentes. Sin embargo, es necesario también tener en cuenta las condiciones de contorno de la ecuación diferencial original, que imponen condiciones sobre los puntos con $i = 0$ y $i = N$. Las condiciones utilizadas se detallaran en la siguiente sección. De este modo la ecuación (2.3) define un sistema de N ecuaciones lineales donde las componentes S_i son las variables a determinar, que en forma matricial puede escribirse como:

$$\mathbf{H}\,\mathbf{S}=-\beta\,\mathbf{V}\,\mathbf{S}, \tag{2.4}$$

donde $\mathbf{S}$ es el vector formado por la función de onda evaluada en las posiciones discretas, $\mathbf{S} = (S_1, S_2, ..., S_N)^T$. La matriz $\mathbf{V}$ está formada por elementos $V_{i,j} = \delta_{i,j}\,V_i$ con $\delta_{i,j}$ la función Delta de Kronecker. Por último la matriz $\mathbf{H}$ es tridiagonal y simétrica, representada como:

$$\mathbf{H}=\begin{pmatrix} d_1 & b_1 & 0 & 0 & \cdots & 0 \\ b_1 & d_2 & b_2 & 0 & \cdots & 0 \\ 0 & b_2 & d_3 & b_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & d_{N-1} & b_{N-1} \\ 0 & 0 & 0 & \cdots & b_{N-1} & d_N \end{pmatrix} \tag{2.5}$$

donde los elementos están definidos directamente por el sistema físico utilizado, siendo de la forma:

$$\begin{aligned} d_i &= \frac{1}{\mu\,dr^2}+\frac{l\,(l+1)}{2\,\mu\,r_i^2}+U_i-E \\ b_i &= -\frac{1}{2\,\mu\,dr^2} \end{aligned} \tag{2.6}$$

para $i = 1, 2, 3, ..., N-1$.

El último elemento de la diagonal está modificado por la condición de contorno y tiene forma similar dada por:

$$d_N = \frac{1}{\mu\, dr^2} + \frac{l\,(l+1)}{2\,\mu\, r_i^2} + U_i - E - \frac{\alpha}{2\,\mu\, dr^2}, \tag{2.7}$$

donde el parámetro α puede ser real o complejo según que condición de borde que se requiera utilizar, como se detallará en la siguiente sección.

La ecuación (2.4) es un problema de autovalores y autovectores generalizado. Por la complejidad numérica que implica la resolución de dicho sistema, se modifica el mismo a fin de llevarlo a un sistema con implementación numérica más simple. Para ello se multiplica ambos lados de la ecuación por $-\mathbf{V}^{-1/2}$ y utilizando $\mathbf{V} = \mathbf{V}^{1/2}\,\mathbf{V}^{1/2}$ se obtiene:

$$\begin{aligned} -\mathbf{V}^{-1/2}\,\mathbf{H}\,\mathbf{S} &= -\,\mathbf{V}^{-1/2}\,\mathbf{H}\,\mathbf{V}^{-1/2}\,\mathbf{V}^{1/2}\,\mathbf{S} \\ &= \mathbf{V}^{-1/2}\,\beta\,\mathbf{V}^{1/2}\,\mathbf{V}^{1/2}\mathbf{S} \\ &= \beta\,\mathbf{V}^{1/2}\,\mathbf{S}. \end{aligned} \tag{2.8}$$

En la ecuación anterior se puede identificar un problema típico de autovalores, redefiniendo las matrices como se muestra a continuación:

$$\begin{aligned} \mathbf{H}' &= -\mathbf{V}^{-1/2}\,\mathbf{H}\,\mathbf{V}^{-1/2} \\ \mathbf{S}' &= \mathbf{V}^{1/2}\,\mathbf{S} \\ \mathbf{V}^{\pm\,1/2} &= Diag(V_1^{\pm\,1/2}, V_2^{\pm\,1/2}, ..., V_N^{\pm\,1/2}) \\ \mathbf{H}'\,\mathbf{S}' &= \beta\,\mathbf{S}' \end{aligned} \tag{2.9}$$

Los elementos de la matriz H' quedan determinados por $d_i' = -\frac{d_i}{V_i}$ en la diagonal. Por otra parte, en los elementos fuera de la diagonal la modificación es otro factor multiplicativo $b_i' = -\frac{b_i}{\sqrt{V_i V_{i+1}}}$ en el elemento i-ésimo.

De esta manera, el problema de autovalores generalizados se transforma en un problema de autovalores usual que puede ser abordado implementando distintos algoritmos. En este trabajo se utilizó el algoritmo llamado iteración QR debido a su eficiencia[7] y a la posibilidad de obtener únicamente una cantidad predeterminada de autovalores de menor magnitud.

2.2. Condiciones de borde

Teniendo en cuenta la necesidad de finalizar la iteración de la ecuación (2.3) se consideran dos condiciones de borde. Para cualquier par de potenciales auxiliar y generador utilizados, las funciones Sturmianas obtenidas deber ser apropiadas para describir el

sistema físico. Por ello, las soluciones de la ecuación de Schrödinger (1.15) deben ser acotadas en todo el espacio, y en particular en el origen. Esto es:

$$\lim_{r \to 0} \frac{S_{n,l}}{r} < \infty. \tag{2.10}$$

Esto implica que la función debe ser necesariamente nula en el origen de coordenadas. En la discretización espacial utilizada, esto se traduce en la nulidad del elemento S_0 en la ecuación de recurrencia (2.4).

Para el borde externo $r > r_c$ se tiene como condición de contorno que la función de onda sea la solución asintótica de la ecuación de Schrödinger (1.15). Considerando que el potencial generador $V(r)$ es de corto alcance r_c y el potencial auxiliar tiende asintóticamente al Coulombiano ($U(r) = \frac{Z}{r}$), puede resolverse la ecuación de Schrödinger correspondiente. Si $F(r)$ es la solución asintótica para el potencial Coulombiano dada por las ecuaciones (1.36) o (1.37), se exige que la función Sturmiana se comporte como $F(r)$ para radios mayores al radio de corte. En la discretización espacial utilizada esto se traduce en decir que $S_N = F(r_N)$.

La iteración en la ecuación (2.4) debe finalizar en S_N ya que para radios mayores al de corte el potencial generador se anula y esto provoca la divergencia de los elementos de la matriz $\mathbf{H}'$. Dado que se conoce como se comportan las funciones Sturmianas para $r > r_c$, se considera a S_{N+1} proporcional a S_N, con constante de proporcionalidad α. Es decir:

$$S_{N+1} = \frac{S_{N+1}}{S_N} S_N = \frac{F(r_{N+1})}{F_{r_N}} S_N = \frac{F((N+1)\,dr)}{F(N\,dr)} S_N = \alpha\, S_N. \tag{2.11}$$

Para energías negativas, la función $F(r)$ está definida por la ecuación (1.37),con lo cual:

$$\alpha = \frac{e^{-(N+1)dr\,\kappa - \frac{Z\,\mu}{\kappa} ln(2\,\kappa\,(N+1)dr)}}{e^{-Ndr\,\kappa - \frac{Z\,\mu}{\kappa} ln(2\,\kappa\,Ndr)}} = e^{-\kappa\,dr - \frac{Z\,\mu}{\kappa} ln(\frac{N+1}{N})} \tag{2.12}$$

mientras que para energías positivas se tienen dos soluciones posibles, una correspondiente a una onda entrante y otra a una saliente. La constante de proporcionalidad para ese caso es:

$$\alpha = \frac{e^{\pm\, i((N+1)dr\,k - \frac{Z\,\mu}{k} ln(2\,k\,(N+1)dr))}}{e^{\pm\, i(Ndr\,k - \frac{Z\,\mu}{k} ln(2\,k\,Ndr))}} = e^{\pm\, i(k\,dr - \frac{Z\,\mu}{k} ln(\frac{N+1}{N}))} \tag{2.13}$$

Dado que la condición de borde depende únicamente de Z, μ y E, todos los elementos de la base encontrada tendrán la misma condición asintótica: todas las funciones Sturmianas oscilarán con el mismo número de onda k. Esta es una característica esencial de las funciones Sturmianas tal como han sido definidas en esta tesis y en trabajos previos[4, 8].

2.3. Algoritmos para el cálculo de los autovalores y autovectores

2.3.1. Determinación de los autovalores

En esta sección se presenta el algoritmo utilizado para la resolución numérica de la ecuación de autovalores (Ec. 2.9), basado en la aplicación sucesiva de transformaciones ortogonales para eliminar los elementos fuera de la diagonal. Supongamos que se desea encontrar los autovalores de una matriz de la forma:

$$\mathbf{H} = \begin{pmatrix} a_1 & b_1 & 0 & \cdots & 0 & 0 \\ b_1 & a_2 & b_2 & \cdots & 0 & 0 \\ 0 & b_2 & a_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & b_{N-2} & a_{N-1} & b_{N-1} \\ 0 & 0 & \cdots & 0 & b_{N-1} & a_N \end{pmatrix}, \tag{2.14}$$

para ello se utiliza el *método QR*, el cual consiste en reemplazar la matriz original por otra semejante, $\mathbf{A} = \mathbf{Q}^T\mathbf{H}\mathbf{Q}$ donde $\mathbf{Q}$ es una matriz ortogonal que satisface la relación $\mathbf{H} = \mathbf{Q}\mathbf{R}$, siendo $\mathbf{R}$ una matriz triangular superior[7]. Por ser $\mathbf{Q}$ ortogonal, la matriz resultante de la transformación tiene los mismos autovalores que la matriz original. Encontrando la transformación que deja diagonal a la matriz $\mathbf{A}$ se pueden utilizar los elementos de la diagonal como autovalores de la matriz original.

El nuevo problema consiste entonces en encontrar la matriz $\mathbf{Q}$ que diagonaliza a $\mathbf{A}$. Esto se realiza aplicando sucesivas rotaciones ortogonales, para ir eliminando progresivamente los elementos fuera de la diagonal. Para esto se aplica una matriz $\mathbf{G}^{(k)}$ que sea la identidad exceptuando en los lugares $G^{(k)}_{k,k}$, $G^{(k)}_{k,k+1}$, $G^{(k)}_{k+1,k}$ y $G^{(k)}_{k+1,k+1}$:

$$\mathbf{G}^{(k)} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 1 & \cdots & \cdots & 0 & 0 \\ \cdots & \cdots & c & s & \cdots & \vdots \\ \cdots & \cdots & -s^* & c & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}, \tag{2.15}$$

llamada *matriz de Givens*. Una iteración del método QR consiste en realizar la transformación $\mathbf{G}^{(k)^T}\,\mathbf{H}\,\mathbf{G}^{(k)}$ para todos los valores $k = 0, 1, ..., N-1$. Se definen los valores de la misma de modo que se satisfaga la relación:

$$\begin{pmatrix} c & s \\ -s^* & c \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sqrt{x_1^2 + x_2^2} \\ 0 \end{pmatrix}. \tag{2.16}$$

Por otro lado, la matriz de Givens debe satisfacer ortogonalidad, esto es:

$$\mathbf{I} = \mathbf{G}\,\mathbf{G}^{\dagger} = \begin{pmatrix} c & s \\ -s^* & c \end{pmatrix} \begin{pmatrix} c^* & -s \\ s^* & c^* \end{pmatrix} = \begin{pmatrix} c\,c^* + s\,s^* & s\,(c^* - c) \\ -s^*(c^* - c) & c\,c^* + s\,s^* \end{pmatrix}. \tag{2.17}$$

De la ecuación anterior se obtiene que $s\,(c^* - c) = 0$ por lo tanto c debe ser puramente real y la suma de los módulos $\|c\|^2 + \|s\|^2$ debe ser la unidad.

Si el vector (x_1, x_2) al que se desea aplicar la matriz de Givens es real, la matriz de rotación será puramente real y se define según el Algoritmo 1.

Algoritmo 1 Algoritmo para obtención de las rotaciones de Givens reales

si $\|x_2\| = 0$ **entonces**
    $c = 1$
    $s = 0$
si no si $\|x_1\| = 0$ **entonces**
    $c = 0$
    $s = f(x_2)$
si no
    $c = \frac{\|x_1\|}{\sqrt{\|x_1\|^2 + \|x_2\|^2}}$
    $s = \frac{x_2}{x_1}\,c$
fin si

Por otro lado, si los elementos (x_1, x_2) son complejos la definición de la matriz de rotación es similar, con algunos cambios en la definición de s. Si el vector es complejo pero posee sólo parte real, el algoritmo funciona de la misma manera que el Algoritmo 1, pero se hace la distinción debido a que el Algoritmo 1 es considerablemente más rápido y es utilizado cuando se conoce de antemano el tipo de variable utilizado. El algoritmo utilizado en el caso complejo es el Algoritmo 2.

En ambos algoritmos se utiliza la función $f(x)$, definida como $f(x) = \frac{x}{\|x\|}$, si x es real la finción es directamente el signo, mientras para el caso complejo es una normalización del argumento. Información más detallada y deducción de los algoritmos anteriores se puede encontrar en [9].

Al aplicar la primera rotación sobre la matriz $\mathbf{H}$ de la forma (2.14) aparece en los sitios $H_{k-1,k+1}, H_{k+1,k-1}$ el valor $\gamma = s\,b_1$. La nueva matriz no es tridiagonal por tener elementos en la segunda diagonal superior e inferior: la matriz resultante es

Algoritmo 2 Algoritmo para obtención de las rotaciones de Givens complejas

si $\|x_2\| = 0$ **entonces**
    $c = 1$
    $s = 0$
si no si $\|x_1\| = 0$ **entonces**
    $c = 0$
    $s = f(x_2)$
si no
    $c = \frac{\|x_1\|}{\sqrt{\|x_1\|^2+\|x_2\|^2}}$
    $s = sg(x_1)\frac{x_2^*}{\sqrt{\|x_1\|^2+\|x_2\|^2}}$
fin si

pentadiagonal. Esto se soluciona al aplicar la siguiente transformación en el mismo sitio, dado que aparece la suma $s\,b_k + c\gamma$. Este elemento puede hacerse nulo eligiendo adecuadamente los valores de c y s según los algoritmos mostrados anteriormente, en este caso, buscando eliminar un vector de la forma $(x_1, x_2) = (-b_k, \gamma)$. Sin embargo cuando se aplica la segunda transformación, se elimina el elemento existente en la pentadiagonal pero aparece uno nuevo que será eliminado con la siguiente rotación. Luego se aplica la siguiente matriz de Givens y así sucesivamente. Cada vez que se utiliza una $\mathbf{G}^{(k)}$ se aumenta $k \to k+1$ para que la siguiente iteración aplique una rotación elimine la pentadiagonal inducida por su predecesor.

Al finalizar de aplicar las N matrices de Givens se obtiene una matriz que sigue siendo simétrica y tridiagonal. Luego la transformación ortogonal completa está dada por el producto de todas las matrices de Givens, esto es:

$$\mathbf{A} = \mathbf{G}^{(N)^T}\mathbf{G}^{(N-1)^T}\cdots\mathbf{G}^{(1)^T}\mathbf{H}\mathbf{G}^{(1)}...\mathbf{G}^{(N-1)}\mathbf{G}^{(N)}, \tag{2.18}$$

de manera tal que el producto de transformaciones es equivalente a una aplicar una única transformación de la forma $\mathbf{Q} = \mathbf{G}^{(1)}...\mathbf{G}^{(N-1)}\mathbf{G}^{(N)}$. Dicha transformación es evidentemente ortogonal por ser producto de transformaciones ortogonales, por ello se utiliza $\mathbf{Q}$ como la matriz correspondiente a la descomposición $\mathbf{QR}$.

A partir de la segunda matriz de Givens ($\mathbf{G}^{(2)}$), se define cada una de ellas para eliminar los elementos fuera de la tridiagonal, sin embargo la primer transformación está indeterminada. Existen varios criterios para elegir la primer matriz, los más utilizados son el *shift de Rayleigh* y el *shift de Wilkinson*[10], sin embargo puede utilizarse cualquier shift pero no está garantizada la convergencia del algoritmo.

Para determinar el shift de Wilkinson se define la primer matriz de Givens utilizando un vector formado por los valores iniciales $x_1 = a_1 - \mu$ y $x_2 = b_1$. El parámetro μ es uno de los autovalores de la última submatriz 2×2 definida por:

$$\mathbf{L}_N = \begin{pmatrix} a_{N-1} & b_{N-1} \\ b_{N-1} & a_N \end{pmatrix} \tag{2.19}$$

y se elige μ como el autovalor más cercano a a_N. Por otra parte, el shift de Rayleigh se define de manera similar pero utilizando $\mu = a_N$. Sin embargo esta elección posee una convergencia más lenta que el shift de Wilkinson[11].

En este trabajo se utiliza el shift de Wilkinson, debido a sus propiedades de convergencia. En efecto, si $b_{N-1}^{(k)}$ es el valor del elemento fuera de la diagonal luego de k iteraciones del método QR, está demostrado[11] que la sucesión $\|b_{N-1}^{(k)}\|$ es monótona decreciente y está asegurada la convergencia de los elementos fuera de la diagonal $\lim_{k\to\infty} \|b_{N-1}^{(k)}\| = 0$. Por otro lado la convergencia es cuadrática en los elementos fuera de la diagonal, esto es, $b_{N-1}^{(k+1)} = \mathcal{O}(b_{N-1}^{(k)^2})$. De este modo, cada iteración del método QR se reduce al Algoritmo 3, donde γ es el elemento que surge fuera de la diagonal al realizar la transformación por $\mathbf{G}^{(k)}$.

Algoritmo 3 Una iteracion del método QR para el cálculo de autovalores

$d = \frac{a_{N-1}-a_N}{2}$
$\mu = a_N - \frac{b_{N-q}^2}{d+sign(d)\sqrt{d^2+b_{N-1}}}$
$x_1 = a_1 - \mu$
$x_2 = b_1$
mientras $k < N$ **hacer**
    $\mathbf{G}^{(k)}$ determinada según el Algoritmo 1 o 2
    $\mathbf{A} = \mathbf{G}^{(k)^T}\mathbf{A}\mathbf{G}^{(k)}$
    si $k < N-1$ **entonces**
        $x_1 = b_{k+1}$
        $x_2 = \gamma$
    fin si
fin mientras

El Algoritmo 3 se realiza de forma iterativa hasta eliminar los elementos fuera de la diagonal, dando lugar al método QR usual, que se describe en el Algoritmo 4. Se utiliza como criterio de corte que los elementos fuera de la diagonal sean suficientemente pequeños comparado a los elementos de la diagonal, esto es, $\|b_i^{(k)}\| < \sqrt{2}(\|a_i^{(k)}\| + \|a_{i+1}^{(k)}\|)\,\epsilon_M$, con $i = 1, 2, 3, ..., N-1$ y ϵ_M el menor valor que cumple $1 + \epsilon_M > 1$.

Una variante implementada del Algoritmo 4 es reducir el tamaño de la matriz cada vez que el último elemento fuera de la diagonal es nulo. De este modo, el costo computacional para calcular el primer autovalor es de orden $\mathcal{O}(N)$, para obtener el segundo autovalor dicho costo se reduce a $\mathcal{O}(N-1)$, y así sucesivamente. El algoritmo

Algoritmo 4 Algoritmo para el cálculo de autovalores manteniendo fijo el tamaño de la matriz (Método QR)

```
1: aux = N
2: mientras aux > N − l hacer
3:     Aplicar Algoritmo 3 a la matriz de tamaño N
4:     si ‖b_{aux−1}‖ < ε_M √2 ‖a_{aux}‖ + ‖a_{aux−1}‖ entonces
5:         aux = aux − 1
6:     fin si
7: fin mientras
```

total para obtener todos los autovalores tiene un costo $\mathcal{O}(\frac{N^2}{2}+N)$. Además, al disminuir el tamaño de la matriz se redefine también el shift de Wilkinson, haciendo que los demás elementos fuera de la diagonal converjan a cero más rápidamente. La principal diferencia es que en la versión original el shift de Wilkinson es siempre el mismo, haciendo rápida la convergencia del último elemento fuera de la diagonal pero lenta la de los demás. El método implementado para este propósito se muestra en el Algoritmo 5.

Algoritmo 5 Algoritmo para el cálculo de autovalores disminuyendo el tamaño de la matriz(Método QR)

```
1: aux = N
2: mientras N > aux − l hacer
3:     Aplicar Algoritmo 3 a la matriz de tamaño N
4:     si ‖b_{N−1}‖ < ε_M √2 ‖a_N‖ + ‖a_{N−1}‖ entonces
5:         N = N − 1
6:     fin si
7: fin mientras
```

Otra variante implementada es utilizar un shift de Wilkinson que busque la convergencia a cero del elemento $b_{N/2}$, utilizando para ello la matriz de 2×2 definida por:

$$\mathbf{L}_{N/2} = \begin{pmatrix} a_{N/2-1} & b_{N/2-1} \\ b_{N/2-1} & a_{N/2} \end{pmatrix}. \tag{2.20}$$

Esto permite separar la matriz, dado que cuando se encuentra la convergencia de $b_{N/2}$, la matriz queda dividida en bloques completamente independientes. Al llegar a esta condición se invoca *recursivamente* al mismo algoritmo para resolver ambas matrices por separado dividiendo el problema hasta llegar a matrices cuya solución es trivial de resolver ($N = 1$ y $N = 2$). El procedimiento se muestra en el Algoritmo 6.

La matriz a la cual se aplica el método QR podría en principio ser reducible, pero

Algoritmo 6 Algoritmo para el cálculo de autovalores recursivamente (Método QR)

si $N = 2$ o $N = 1$ **entonces**
    Resolver caso trivial
    Terminar QR
fin si
mientras $b_{\frac{N}{2}-1} < \epsilon_M \sqrt{2}\,\|a_{\frac{N}{2}-1}\| + \|a_{\frac{N}{2}}\|$ **hacer**
    $d = \frac{a_{N/2-1} - a_{N/2}}{2}$
    $\mu = a_N - \frac{b_{N/2-1}^2}{d + sign(d)\sqrt{d^2 + b_{N/2-1}}}$ ▷ Defino el shift de Wilkinson usando $L_{N/2}$
    $x_1 = a_1 - \mu$
    $x_2 = b_1$
    mientras $k < N$ **hacer**
        $\mathbf{G}^{(k)}$ determinada según el Algoritmo 1 o 2
        $\mathbf{A} = \mathbf{G}^{(k)^T}\mathbf{A}\mathbf{G}^{(k)}$
        si $k < N - 1$ **entonces**
            $x_1 = b_{k+1}$
            $x_2 = \alpha$
        fin si
    fin mientras
fin mientras
aplicar Algoritmo 6 para matriz entre 0 y $N/2 - 1$
aplicar Algoritmo 6 para matriz entre $N/2$ y N

si al menos un elemento fuera de la diagonal es nulo la matriz sería separable en dos bloques y ambos pueden ser resueltos de forma totalmente independiente. Esto implica una cantidad considerablemente menor de cálculos. Mientras que en una matriz de tamaño N el cálculo de todos los autovalores implica una cantidad de operaciones de orden $\mathcal{O}(N^2)$, una separada en bloques de tamaño N_1 y N_2 ($N_1 + N_2 = N$) implica una cantidad de operaciones de orden $\mathcal{O}(N_1^2 + N_2^2)$. Más aún, si se desean los últimos m autovalores de la matriz, la cantidad de operaciones sería sólo de orden $\mathcal{O}(m\,N_2)$, frente a las $m\,N$ operaciones de la matriz original.

Para aprovechar las ventajas de las matrices de bloques se implementó una búsqueda sobre todos los elementos fuera de la diagonal para asegurar que la matriz sea irreducible, en caso de tener algún elemento que cumpla con el criterio de convergencia se divide la matriz en dos bloques independientes. Luego se aplica una iteración de QR y se busca nuevamente la irreducibilidad de la matriz.

Siempre que se encuentra un tamaño en el cual la matriz es reducible se resuelve primeramente la matriz que comienza en el punto donde se encontró el elemento nulo. De modo general, si la matriz posee tamaño N y se encuentra un elemento nulo en N_2, la nueva matriz será la que comienza en $N_2 + 1$ y termina en N, de este modo la nueva matriz tendrá un tamaño inferior a la original y la cantidad de iteraciones para eliminar el último elemento será consecuentemente menor.

Por otro lado, si el último elemento es nulo o satisface a condición de convergencia simplemente se disminuye el tamaño de la matriz en uno y se continúa iterando. El algoritmo implementado para tener en cuenta la posibles reducciones del tamaño de la matriz se detallará más adelante (Algoritmo 11).

Si bien la reducción en dos bloques independientes permite la paralelización del algoritmo resolviendo ambas matrices al mismo tiempo, en el caso de interés esto no es necesario ya que las bases utilizadas son las asociadas a autovalores de menor magnitud. Las matrices características que definen las bases Coulombianas poseen la propiedad de que, por como está construida la matriz; cuando se encuentra un autovalor a través de la aplicación del método QR, este será el de menor módulo. Por ello no es necesario iterar hasta encontrar todos los autovalores y por lo tanto no es necesario paralelizarlo totalmente.

2.3.2. Método de la potencia inversa

Una vez resuelto el problema de encontrar los autovalores de la matriz es necesario buscar los autoestados asociados. Entre los algoritmos iterativos para calcular autovectores, el método más directo es el llamado *método de la potencia*[7], consiste en aplicar sucesivamente el operador $\mathbf{H}$ sobre un vector normalizado $\mathbf{q}^{(k)}$. De modo que $\mathbf{q}^{(k+1)} = \mathbf{H}\,\mathbf{q}^{(k)}$

Dicho algoritmo asegura la convergencia de la iteración, obteniéndose el autovector asociado al autovalor de módulo máximo. La desventaja radica en que el método requiere que los autovalores de módulo máximo sean únicos. El vector inicial $\mathbf{q}^0$ se puede representar como combinación lineal de los autovectores ($\mathbf{v}_i$) de $\mathbf{H}$, esto es:

$$\mathbf{q}^0 = \sum_{i=0}^{N} a_i\, v_i \tag{2.21}$$

Entonces, si se multiplica este vector por la matriz $\mathbf{H}$ m veces, se tiene que:

$$\begin{aligned} \mathbf{H}^m\,\mathbf{q}^0 &= \sum_{i=0}^{N} a_i\,\mathbf{H}^m\,\mathbf{v}_i \\ &= \sum_{i=0}^{N} a_i\,\lambda_i^m\,\mathbf{v}_i \end{aligned} \tag{2.22}$$

con λ_i los autovalores de $\mathbf{H}$. De lo anterior se puede ver que el algoritmo luego de un número de iteraciones determinado por los coeficientes a_i y el modulo de los autovalores, convergerá al autovalor de mayor módulo. Hay que notar que el algoritmo no converge si el vector $\mathbf{q}^0$ es ortogonal al autovector deseado.

En el caso de interés (cálculo de Sturmianas) se desea un conjunto de autovectores y en general asociado a los autovalores de menor módulo. Por ello se utiliza un método muy similar, basado también en la obtención el autovector asociado al autovalor de modulo máximo. El método, llamado *método de la potencia inversa*, consiste en aplicar reiteradas veces el método de la potencia con una matriz $\mathbf{A}$ dada por:

$$\mathbf{A} = (\mathbf{H} - \lambda\,\mathbf{I})^{-1}. \tag{2.23}$$

Si λ_i es un autovalor de la matriz $\mathbf{H}$, entonces $\lambda_i - \lambda$ es autovalor de $\mathbf{H} - \lambda\,\mathbf{I}$ asociado al mismo autovector. Por último $(\lambda_i - \lambda)^{-1}$ es autovalor de $\mathbf{A}$ definido según (2.23) y nuevamente, asociado al mismo autovector. Si se quiere obtener un autovector asociado a un autovalor λ_l basta elegir la semilla λ de modo que se cumpla que:

$$(\lambda_l - \lambda)^{-1} > (\lambda_i - \lambda)^{-1} \qquad \forall i \neq l. \tag{2.24}$$

De este modo, el método de la potencia aplicado directamente sobre la matriz $\mathbf{A}$ converge directamente al autovector deseado[7].

El procedimiento para la obtención de autovectores mediante el método de la potencia inversa se muestra en el Algoritmo 7.

Algoritmo 7 Algoritmo para la obtención de los autovectores (Método de la Potencia Inversa)

1: $k = 1$
2: **mientras** $\|\mathbf{q}^{(k)} - \mathbf{q}^{(k-1)}\| > \epsilon$ **hacer**
3: $\quad \mathbf{y}^{(k)} = (\mathbf{H} - \lambda\,\mathbf{I})^{-1}\,\mathbf{q}^{(k-1)}$
4: $\quad \mathbf{q}^{(k)} = \mathbf{y}^{(k)} / \|\mathbf{y}^{(k)}\|_2$
5: $\quad \lambda = \mathbf{q}^{(k)^T}\mathbf{H}\mathbf{q}^{(k)}$
6: $\quad k = k + 1$
7: **fin mientras**

En cada iteración, se puede calcular nuevamente el valor λ, y este tendrá un valor más próximo al exacto. Esto es opcional para el cálculo de autovectores cuando las condiciones de borde son reales, sin embargo para el caso de autovalores complejos será estrictamente necesario realizar la corrección del autovalor. En nuestro caso de interés, el algoritmo deja de converger luego de un determinado número de iteraciones y es necesario perturbar la matriz para que continúe convergiendo.

Puede demostrarse[7] que para un dado vector inicial $\mathbf{q}^{(0)}$, existe un θ_k entre 0 y π, tal que $\cos\theta_k = \|\mathbf{q}_1^T\,\mathbf{q}^{(k)}\|$, con $\mathbf{q}_1$ el autovector asociado al autovalor de máximo modulo. Dicho θ_k cumple además la relación:

$$\cos\theta_k = \sqrt{1 - tg^2\theta_0 \|\frac{\lambda_1 - \lambda}{\lambda_2 - \lambda}\|^{2\,k}} \tag{2.25}$$

donde λ_1 es el autovalor más cercano a la semilla λ, mientras λ_2 es el segundo más cercano. ésto permite ver la relación entre la velocidad de convergencia del método y la semilla utilizada. Del análisis anterior se deduce que el algoritmo nunca converge si la matriz utilizada tiene autovalores iguales, o cuya distancia a la semilla sea la misma. Así mismo el algoritmo tampoco converge si el vector semilla $q^{(0)}$ es ortogonal al autovector buscado, ya que si $cos(\theta_0) = \|\mathbf{q}_1^T \mathbf{q}^{(0)}\| = 0$, la tangente está indeterminada.

Para asegurar la convergencia del método al autovector deseado se utilizó como semilla $\lambda = \lambda_i$ aprovechando que el autovalor encontrado con el método QR es una aproximación suficientemente buena.

De este modo sólo resta realizar la multiplicación dada por:

$$\mathbf{y} = (\mathbf{H} - \lambda\,\mathbf{I})^{-1}\,\mathbf{q} \tag{2.26}$$

en el Algoritmo 7. Para ello se modifica levemente la ecuación para evitar calcular la matriz inversa debido al alto costo computacional que posee y se busca la solución de la ecuación $(\mathbf{H}-\lambda\,\mathbf{I})\,\mathbf{y} = \mathbf{q}$. La nueva ecuación puede ser resuelta por numerosos métodos, en este trabajo se utilizó la *descomposición LU* y la *Reducción Cíclica* o CR por sus siglas en inglés. A continuación se detalla como se implementaron los algoritmos.

2.3.3. Descomposición LU

La descomposición LU de una matriz dada $\mathbf{A}$ consiste en buscar matrices triangulares superiores ($\mathbf{U}$) e inferiores ($\mathbf{L}$), que cumplan la relación $\mathbf{LU} = \mathbf{A}$. En el caso de interés se tiene la matriz $\mathbf{A} = \mathbf{H} - \lambda\,\mathbf{I}$.

El sistema de ecuaciones se resuelve utilizando las matrices obtenidas de la descomposición, esto es:

$$(\mathbf{H} - \lambda\,\mathbf{I})\,\mathbf{y} = \mathbf{L}\,\mathbf{U}\,\mathbf{y} = \mathbf{q} \tag{2.27}$$

definiendo el problema intermedio de calcular el vector $\mathbf{z}$ que satisface $\mathbf{q} = \mathbf{L}\,\mathbf{z}$, luego la ecuación (2.27) se reduce a encontrar el vector $\mathbf{y}$ que cumpla la relación $\mathbf{z} = \mathbf{L}\,\mathbf{y}$.

Los dos sistemas de ecuaciones que deben resolverse para obtener $\mathbf{y}$, pueden implementarse fácilmente debido a la forma simple de la descomposición. Es interesante notar que las matrices $\mathbf{L}$ y $\mathbf{U}$ tendrán elementos no nulos sólo en la diagonal y en la diagonal inferior y superior respectivamente.

Los elementos pueden encontrarse facilmente partiendo de la forma que tendrán las matrices $\mathbf{L}$ y $\mathbf{U}$

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & & & \\ l_2 & 1 & & & \\ & \ddots & & & \\ & & & 1 & 0 \\ & & & l_N & 1 \end{pmatrix} \text{ y } \mathbf{U} = \begin{pmatrix} u_1 & a_1 & & & \\ 0 & u_2 & a_2 & & \\ & & \ddots & & \\ & & & u_{N-1} & a_{N-1} \\ & & & 0 & u_N \end{pmatrix} \tag{2.28}$$

donde a_i son los elementos fuera de la diagonal de la matriz original, mientras u_i y l_i se obtienen mediante el Algoritmo 8.

Algoritmo 8 Algoritmo para la Descomposición LU

1: $u_1 = A_{1,1}$
2: **mientras** $i < N$ **hacer**
3: $\quad l_i = \frac{A_{i,i-1}}{u_{i-1}}$
4: $\quad u_i = A_{i,i} - l_i\, A_{i-1,i}$
5: **fin mientras**

Dado que la matriz es diagonal dominante ($\|A_{i,i}\| > \|A_{i,i+1}\| + \|A_{i+1,i}\|$), no es necesario implementar la descomposición utilizando pivoteo. La implementación de la descomposición implica una cantidad de iteraciones de orden $\mathcal{O}(N)$, asimismo, la solución de la ecuación (2.27) implica un número similar de iteraciones.

El algoritmo implementado para resolver la ecuación (2.27) consiste en una eliminación gaussiana y se muestra en el Algoritmo 9.

Algoritmo 9 Algoritmo para la resolución de $\mathbf{L}\,\mathbf{U}\,\mathbf{y} = \mathbf{q}$

1: **mientras** $i < N$ **hacer**
2: $\quad q_i = q_i - l_i\, q_{i-1}$
3: **fin mientras**
4: $q_{N-1} = \frac{q_{N-1}}{u_{N-1}}$
5: $i = N - 1$
6: **mientras** $i > 0$ **hacer**
7: $\quad y_i = \frac{(q_i - q_{i+1}\, a_i)}{u_i}$

El Algoritmo 9 así como la descomposición LU dada en el Algoritmo 8 se implementan de la misma manera tanto para reales como para complejos. En el Algoritmo 9 se sobreescribe el vector inicial q dejando el resultado de la operación en el mismo.

Es importante notar que la descomposición **LU** se realiza una única vez para cada λ de interés cuando los elementos son reales, sin embargo para autovalores complejos se debe realizar la corrección al autovalor y volver a realizar la descomposición LU. Luego sobre la descomposición ya encontrada se itera hasta lograr la convergencia del

método de la potencia inversa. Si bien tanto la descomposición $\mathbf{L}\,\mathbf{U}$ como la solución del sistema (2.27) es de primer orden con el tamaño del mismo, el número de veces que debe iterarse sobre el método de la potencia está indeterminado. Esto es fuertemente dependiente tanto de la semilla utilizada para el autovalor como del vector inicial ($\mathbf{q}^0$).

2.3.4. Reducción cíclica

Otro algoritmo utilizado para la resolución de la ecuación $(\mathbf{H} - \lambda\,\mathbf{I})\mathbf{y} = \mathbf{q}$ es la reducción cíclica o *Cyclic Reduction* (CR) por sus siglas en inglés. Este algoritmo es ampliamente utilizado para resolver sistemas tridiagonales[12] debido a la fácil implementación del algoritmo serial y principalmente por la posibilidad de implementar el algoritmo de forma paralela (Detalles de esta implementación se encontrarán en la próxima sección).

El algoritmo consiste en realizar operaciones por filas a fin de generar un sistema de menor tamaño que el original, partiendo de un sistema correspondiente a una matriz tridiagonal con elementos b_i en la diagonal mientras a_i y c_i son los elementos fuera de la diagonal en la fila i-ésima. Esto forma un sistema de N ecuaciones de la forma:

$$a_i\,x_{i-1} + b_i\,x_i + c_i\,x_{i+1} = q_i, \tag{2.29}$$

donde la iteración finaliza con las condiciones de borde $a_{-1} = 0$ y $c_N = 0$, la variable x_i depende directamente de las variables x_{i-1} y x_{i+1}. Si a la fila i-ésima (f_i) se le suma la siguiente fila f_{i+1} multiplicada por una constante β, y la precedente f_{i-1} multiplicada por γ, esto es $f_i = f_i - \gamma\,f_{i-1} - \beta\,f_{i+1}$, la fila resultante de la transformación es:

$$a_i''\,x_{i-2} + a_i'\,x_{i-1} + b_i'\,x_i + c_i'\,x_{i+1} + c''\,x_{i+2} = d_i', \tag{2.30}$$

donde:

$$\begin{aligned}
a_i'' &= \gamma\,a_{i-1} \\
c_i'' &= \beta\,c_{i+1} \\
a_i' &= a_i + \gamma\,b_{i-1} \\
c_i' &= c_i + \beta\,b_{i+1} \\
b_i' &= b_i + \gamma\,c_{i-1} + \beta\,a_{i+1} \\
d_i' &= d_i + \gamma\,d_{i-1} + \beta\,d_{i+1}
\end{aligned} \tag{2.31}$$

Se puede observar que definiendo $\gamma = -a_i/b_{i-1}$ y $\beta = -c_i/b_{i+1}$, las constantes de (2.32) se simplifican a sólo tres de ellas, (a_i'', b_i' y c_i'') con $a_i'' = \gamma\,a_{i-1}$, $b_i' = b_i + \gamma\,c_{i-1} + \beta\,a_{i+1}$ y $c_i'' = \beta\,c_{i+1}$. De este modo, si se aplica esta transformación sobre todas las filas

pares se obtiene un sistema de $N/2$ ecuaciones de la forma:

$$a''_i\, x_{i-2} + b'_i\, x_i + c''_i\, x_{i+2} = d'_i \tag{2.32}$$

con las condiciones de borde $a''_{-1} = a''_{-2} = 0$ y $d''_N = d''_{N+1} = 0$. Así, las filas pares dependen sólo de las variables pares. Esta independencia permite modificar el problema en uno reducido donde se debe encontrar las soluciones de la matriz formada sólo por las filas pares, que tendrá la mitad de incógnitas. Luego de obtener la solución para las variables de índice par, se utilizan las ecuaciones restantes formadas por las filas impares para obtener las variables impares, esto implica:

$$a_i\, x_{i-1} + b_i\, x_i + c_i\, x_{i+1} = d'_i, \tag{2.33}$$

para índices impares. De este modo, una vez obtenidas todas las variables pares, se obtienen directamente las impares. Por la semejanza entre las ecuaciones (2.32) y (2.29) se puede observar que la matriz resultante de considerar sólo las filas pares es tridiagonal. Por ello se redefine la matriz formada por las filas y columnas pares para que sus índices se muevan entre 0 y $N/2$ (la fila i-ésima de la matriz original, ahora se corresponde con la fila (i/2)-ésima) y se vuelve a aplicar el algoritmo CR. Se deja de dividir y redefinir la matriz cuando se llega al caso trivial de tamaño $N = 2$, allí se resuelve trivialmente y se recupera el valor de las variables impares. Luego se amplía la matriz y se recuperan las variables impares de la nueva matriz.

Para fijar ideas, se muestra a continuación como se realiza la reducción cíclica sobre una matriz de tamaño 5×5, en todos los casos se considera a la primer fila de la matriz como la fila 0 y por lo tanto, par. Se considera la matriz dada por:

$$\begin{pmatrix} b_0 & c_0 & 0 & 0 & 0 \\ a_1 & b_1 & c_1 & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 \\ 0 & 0 & a_3 & b_3 & c_3 \\ 0 & 0 & 0 & a_4 & b_4 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix} \tag{2.34}$$

aplicando el algoritmo sobre las filas pares, se obtiene el sistema:

$$\begin{pmatrix} b'_0 & 0 & c''_0 & 0 & 0 \\ a_1 & b_1 & c_1 & 0 & 0 \\ a''_2 & 0 & b'_2 & 0 & c''_2 \\ 0 & 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a''_4 & 0 & b'_4 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} d'_0 \\ d_1 \\ d'_2 \\ d_3 \\ d'_4 \end{pmatrix} \tag{2.35}$$

En la matriz resultante (2.35), las filas pares dependen únicamente de las variables

pares. De este modo se puede separar una matriz de menor tamaño formado por las filas y columnas pares, se obtiene la ecuación matricial:

$$\begin{pmatrix} b'_0 & c''_0 & 0 \\ a''_2 & b'_2 & c''_2 \\ 0 & a''_4 & b'_4 \end{pmatrix} \begin{pmatrix} x_0 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} d''_0 \\ d'_2 \\ d'_4 \end{pmatrix} \tag{2.36}$$

Para simplificar la notación, se redefinen los elementos de la matriz (2.35) como $a''_i = \alpha_i$, $b'_i = \beta_i$ y $c''_i = \gamma_i$. De este modo, la matriz equivalente formada por los elementos nuevos se puede volver a simplificar mediante la reducción cíclica utilizando las transformaciones (2.32). Así se obtiene la matriz equivalente:

$$\begin{pmatrix} \beta'_0 & 0 & \gamma''_0 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha''_4 & 0 & \beta'_4 \end{pmatrix} \begin{pmatrix} x_0 \\ x_2 \\ x_4 \end{pmatrix} = \begin{pmatrix} \delta'_0 \\ \delta_2 \\ \delta'_4 \end{pmatrix} \tag{2.37}$$

La ecuación anterior forma un sistema de 2×2 con las filas múltiplo de cuatro cuya solución es trivial:

$$\begin{pmatrix} \beta'_0 & \gamma''_0 \\ \alpha''_4 & \beta'_4 \end{pmatrix} \begin{pmatrix} x_0 \\ x_4 \end{pmatrix} = \begin{pmatrix} \delta'_0 \\ \delta'_4 \end{pmatrix} \implies \begin{pmatrix} x_0 \\ x_4 \end{pmatrix} = \frac{1}{\beta'_0\,\beta'_4 - \gamma''_0\,\alpha''_4} \begin{pmatrix} \beta'_4\,\delta'_0 - \gamma''_0\,\delta'_4 \\ \beta_0\,\delta'_4 - \alpha''_4\,\delta'_0 \end{pmatrix} \tag{2.38}$$

Cuando ya se conocen las variables x_4 y x_0, utilizando la matriz (2.37) se puede encontrar el valor de x_2 ya que se cumple que $\alpha_2\,x_0 + \beta_2\,x_2 + \gamma_2\,x_4 = \delta_2$.

Por último, una vez obtenido x_2 se puede recuperar de la matriz (2.34) los valores de x_1 y x_3, despejándolo de las ecuaciones $a_1\,x_0 + b_1\,x_1 + c_1\,x_2 = d_1$ y $a_3\,x_2 + b_3\,x_3 + c_3\,x_4 = d_3$ correspondientes a las filas 1 y 3 respectivamente.

El algoritmo implementado para resolver una matriz tridiagonal de tamaño $N \times N$ se presenta a continuación:

Algoritmo 10 Algoritmo para Reducción cíclica

```
1: delta = 1
2: iters = log2(N)
3: mientras i < iters hacer
4:     Aplicar operaciones por filas a las filas multiplo de 2 delta
5:     delta = delta * 2
6: fin mientras
7: Resolver caso trivial 2x2
8: mientras i < iters hacer
9:     Recupero variables impares
10:    delta = delta/2
11: fin mientras=0
```

2.4. Optimización de la obtención de la bases Sturmianas

El objetivo principal de este trabajo es optimizar el cálculo de bases Sturmianas. Éste consta de dos partes principales, la obtención de autovalores y la determinación de los autovectores asociados. Para optimizar algoritmos se planteó utilizar procesadores gráficos (GPU por sus siglas en inglés, *Graphics Processor Unit*), utilizados principalmente para el procesamiento de gráficos y para los videojuegos debido a la gran cantidad de núcleos que posee. Sin embargo en los últimos años comenzaron a ser utilizadas para propósitos generales (GPGPU por sus siglas en inglés, *General Purpose Computing on Graphics Processing Units*). Entre las aplicaciones desarrolladas para GPGPU se encuentran las simulaciones del clima y de fluidos, finanzas, inteligencia artificial y *deep learning*, procesamiento de imágenes y señales, entre otros problemas con alto grado de paralelismo.

La GPU está formada por un conjunto de multiprocesadores (SM por sus siglas en inglés, *Streaming Multiprocessors*). Dentro de un multiprocesador se ejecutan bloques de hilos de código. Debido a la arquitectura del hardware, los bloques son ejecutados en un orden arbitrario para asegurar escalabilidad, y por lo tanto, deben ser completamente independientes. Por ejemplo, si una GPU puede ejecutar cuatro bloques al mismo tiempo y por conveniencia se utilizan ocho, entonces hay cuatro bloques inactivos esperando a que se desocupe algún multiprocesador. La propiedad de ejecutar los bloques independientemente permite al programador asumir que se poseen tantos bloques como sea necesario.

Para poder manejar grandes cantidades de hilos los multiprocesadores utilizan una arquitectura SIMT (del inglés, *Single Instruction, Multiple-Threads*). Cuando un programa lo solicita el multiprocesador crea y organiza los hilos en conjuntos de 32 unidades, llamados *warps*. Todos los hilos dentro de un mismo warp deben ejecutar las mismas instrucciones, por ello tienen un *warp scheduler* que decide que conjunto de instrucciones realizan determinados hilos de un mismo warp. En el mejor caso todos los hilos realizan el mismo trabajo, pero en el caso de haber alguna ramificación en el código se ejecutan serialmente.

Para la implementación de los algoritmo que se ejecuta en la GPU se utilizó CUDA[13], un extensión de C desarrollada por NVIDIA para la programación funciones (también llamadas *kernel*) que pueden ser ejecutadas en la GPU.

El paradigma de programación utilizado consiste en una gran cantidad de hilos que pueden ser ejecutados en paralelo. Los hilos se agrupan en bloques que pueden ser unidimensionales, bidimensionales o tridimensionales, dentro de cada bloque los hilos poseen un identificador único con el cual se controla qué parte del código realiza cada

uno. Como ejemplo simple, en CUDA sumar dos vectores se convierte en una linea de código, indicando al hilo i-ésimo que sume los elementos i-ésimos de un vector a otro.

Los hilos de un mismo bloque comparte una memoria de acceso rápido pero de tamaño reducido, llamada memoria compartida (*shared memory*). Por otro lado, todos los hilos tienen acceso a una memoria global, más lenta pero visible por todos. Las variables definidas dentro de un kernel son almacenadas en la memoria de registros. Esta memoria es compartida por todos los hilos de un bloque, por ello el tamaño de un bloque está limitado por la cantidad de registros utilizados por cada hilo.

En el caso de las bases Sturmianas, se pueden identificar dos algoritmos esencialmente diferentes: el cálculo de autovalores y el de autovectores. El primero es serial y cada iteración del algoritmo es completamente dependiente de la iteración anterior. Más aún, cada rotación de Givens depende de la anterior para definir correctamente los elementos c y s. Los algoritmos existentes para cálculo de autovalores en GPU encuentran todos los autovalores, mientras que sólo los de menor módulo son utilizados para el cálculo de Sturmianas. Algunas librerías tienen desarrolladas funciones que calculan un número reducido de autovalores pero sólo están implementadas para matrices simétricas con elementos reales o bien para complejas que sean hermíticas. La matriz utilizada posee un único elemento complejo fuera de la diagonal debido a las condiciones de borde que hacen que la matriz no se hermítica y por lo tanto no cumpla ninguna de las condiciones requeridas. s El segundo de los métodos, el cálculo de los autovectores, puede ser paralelizado e implementado de modo que se resuelva asincrónicamente mientras la CPU continúa obteniendo autovalores. De esta manera, cada vez que se encuentra un autovalor, se lanza el cálculo en la GPU. Un esquema simplificado del algoritmo para una matriz (M) de tamaño N a la cual se desea calcular los primeros l autovalores se muestra en el Algoritmo 11.

2.5. Paralelización del algoritmo CR

Bajo la idea de utilizar muchos hilos con la misma instrucción es posible paralelizar la operación por filas del algoritmo CR, por ejemplo, que todos los hilos pares transformen las filas pares, o que el hilo i-ésimo transforme la fila $2\,i$ para no usar el doble de hilos que lo necesario. Por ejemplo, si se dispone de 1024 hilos, en una matriz de tamaño 30000, entonces cada hilo realizará transformaciones en la fila par correspondiente. De todos modos, esto no es suficiente para reducir la matriz completa, entonces el hilo i-ésimo transformará las filas $2\,i$, $2\,i + 2048$, $2\,i + 4096$ y continuando hasta que $2\,i + n\,2048$ sea mayor a 30000, para cierto n. En esta instancia la matriz de interés se redujo a la mitad, por ello ahora el hilo i-ésimo aplica transformaciones sobre

Algoritmo 11 Algoritmo para el cálculo de autovalores y autovectores en simultaneo

1: $min = 0$
2: **mientras** $N > N - l$ **hacer**
3: **si** $N \leq min$ **entonces**
4: $N = min - 1$
5: $min = 0$
6: **fin si**
7: Aplico Algoritmo 3 a la matriz entre fila min y N
8: **si** $\|M_{N,N-1}\| < \sqrt{2}\,\epsilon_M\,(\|M_{N,N}\| + \|M_{N-1,N-1}\|)$ **entonces**
9: **si** USAR GPU **entonces**
10: Lanzar asincrónicamente el cálculo del autovector en la GPU
11: **si no**
12: Lanzar cálculo de autovector con Algoritmo 7 en CPU
13: **fin si**
14: $N = N - 1$
15: **fin si**
16: $j = N$
17: **mientras** $j \geq min$ **hacer**
18: **si** $\|M_{N,N-1}\| < \sqrt{2}\,\epsilon_M\,(\|M_{N,N}\| + \|M_{N-1,N-1}\|)$ **entonces**
19: $min = j + 1$
20: continuar en linea 7
21: **fin si**
22: **fin mientras**
23: **fin mientras**

las filas $4\,i$, $4\,i + 4096$, $4\,i + 8192$ y continuando según sea necesario. Luego se hace lo mismo sobre las filas múltiplos de ocho, dieciséis, treinta y dos, y así hasta llegar al caso trivial. Para obtener el resto de las variables se realiza el proceso inverso como se explicó en la Sección 2.3.4.

Existen numerosos métodos numéricos optimizados para el funcionamiento en la GPU, por ejemplo entre el conjunto de funciones implementadas en *cuSPARSE*[14] se encuentra la función `cusparseZgtsv_nopivot` que resuelve la ecuación $\mathbf{Ax} = \mathbf{b}$ para matrices tridiagonales de forma asincrónica utilizando el algoritmo de reducción cíclica. El problema principal de utilizar dicha función, es que es necesario que el cálculo de autovalores sea completamente asincrónico, todo el Algoritmo 7 debe ejecutarse dentro de un mismo kernel. El algoritmo implementado para dicho propósito se muestra en el Algoritmo 12, que sintetiza el procedimiento descripto en los parrafos anteriores.

Algo influyente a la hora de implementar códigos en CUDA es la utilización de memoria compartida, por su gran velocidad frente a la memoria global. En la GPU en general esta memoria es restringida, siendo 96Kb las memorias compartidas más grandes actualmente. En el algoritmo implementado no se pudo utilizar memoria compartida debido al gran tamaño de la matriz, ocupando alrededor de 600Kb para $N = 30000$ y elementos tipo *float*. Por ello se implementó utilizando memoria global.

Algoritmo 12 Algoritmo para la realizar la reducción cícica (CR) en la GPU

1: id =Identificador del hilo
2: $iter = log_2(N)$
3: $aux = N/2$
4: $delta = 1$
5: Inializar vector normalizado **d**
6: **mientras** $i < iter$ **hacer**
7: **mientras** $id < aux$ **hacer**
8: $indice = id$*$2 * delta$
9: $imayor = indice + delta$
10: $imenor = indice - delta$
11: Transformar linea "indice" utilizando "imayor" e "imenor"
12: $id = id$+Num_total_hilos
13: **fin mientras**
14: Sincronización
15: $aux = aux/2$
16: $delta = 2$*$delta$
17: **fin mientras**
18: Resolver caso 2×2
19: **mientras** $i < iter$ **hacer**
20: **mientras** $id < aux$ **hacer**
21: $indice = id$*$delta + delta/2$
22: $imayor = indice + delta/2$
23: $imenor = indice - delta/2$
24: Recupero variable de fila "indice", usando las filas "imayor" e "imenor"
25: $id = id$+Num_total_hilos
26: **fin mientras**
27: Sincronización
28: $delta = delta/2$
29: $aux = aux$*2
30: **fin mientras**

2.6. Cálculo del autovector en GPU

Para que el cálculo del autovector sea completamente asincrónico es necesario implementar el método de la potencia inversa (Algortmo 7) completo dentro de un solo kernel. En el algoritmo se pueden identificar varias partes independientes, la más importante es la resolución del sistema $(\mathbf{H} - \lambda\,\mathbf{I})\,\mathbf{y} = \mathbf{q}$ que se realiza con el Algoritmo 12 presentado anteriormente.

La otra parte del algoritmo consiste en normalizar el vector, evaluar la convergencia y calcular el nuevo valor de λ. Cada uno de estos tres problemas consiste en una *reducción*, que consiste en sumar todos los elementos de un vector (**aux**) de tamaño N y guardarlo en el primer elemento. La implementación en CUDA consiste en que los primeros $N/2$ hilos realicen la suma $aux_i = aux_i + aux_{2i}$ donde i es el identificador del hilo. Ahora falta hacer la reducción de la primer mitad de **aux**, entonces los primeros $N/4$ realizan la misma suma $aux_i = aux_i + aux_{2i}$, y el falta reducir el primer cuarto de vector, continuando hasta llegar a un vector de tamaño unidad. Una representación del procedimiento se encuentra en el Algoritmo 13, y un ejemplo simple se muestra en la Figura (2.1).

Algoritmo 13 Suma por reducción de un vector (**aux**) de tamaño N en la GPU

```
1: i =Identificador del hilo
2: N = N/2
3: mientras N > 1 hacer
4:     si i < N entonces
5:         aux_i = aux_i + aux_{2i}
6:         N = N/2
7:     fin si
8:     Sincronización de hilos
9: fin mientras
```

Para las reducciones realizadas se utilizó un vector (**aux**) definido en la memoria compartida por que el tamaño lo permite.

Para verificar la convergencia se puede utilizar un vector donde se guarda en lugar i-ésimo la diferencia entre el vector (en el lugar i-ésimo) antes y después de la iteración, esto se muestra en el Algoritmo 14, donde ϵ es la tolerancia en el cálculo, **aux** es el vector sobre el cual se realiza la reducción y $\mathbf{q}^k$ es el autovector despues de k iteraciones del método de potencia inversa. Si el resultado luego de la reducción es distinto de cero, significa que al menos un elemento del autovector aún no convergió.

Para normalizar el vector el procedimiento es similar, almacenando en el vector **aux** el cuadrado del valor del autovector $\mathbf{q}^k$, esto es: $aux_i = \|q_i^k\|^2$. Luego realizando la reducción se tiene en el primer elemento el módulo al cuadrado del vector $\mathbf{q}^k$. Para

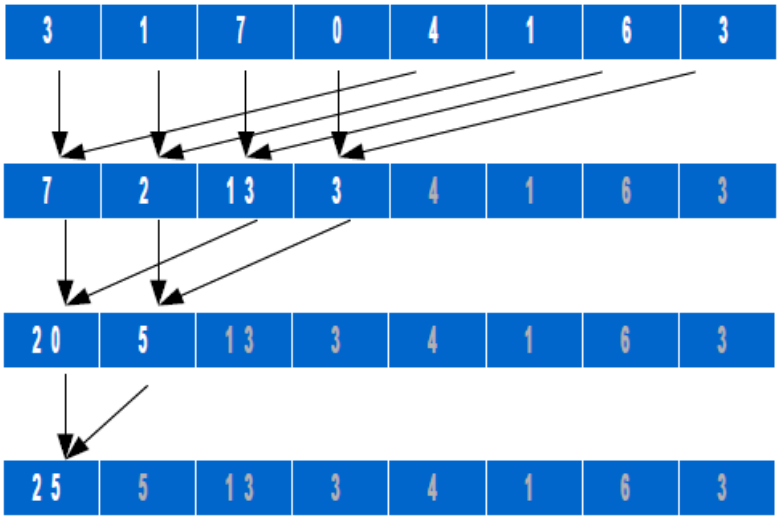

Figura 2.1: Ejemplo del algoritmo de reducción para un vector de ocho elementos.

el nuevo valor de λ se guarda en el vector el valor $aux_i = q_i\,(a_i\,q_{i-1} + b_i\,q_i + c_i\,q_{i+1})$ con a_i, b_i y c_i los elementos de la diagonal inferior, diagonal y diagonal superior de la fila i-ésima respectivamente. Al realizar la reducción se tendrá en el primer elemento de **aux** el nuevo autovalor.

Algoritmo 14 Determinación de la convergencia del método de potencia inversa

1: i =Identificador del hilo
2: **si** $\|q_i^k - q_i^{k-1}\| < \epsilon$ **entonces**
3: $\quad aux_i = 1$
4: **si no**
5: $\quad aux_i = 0$
6: **fin si**

Teniendo en cuenta lo expuesto en los parrafos anteriores, el algoritmo completo para el cálculo del autovector empieza con definir un vector inicial, continúa con el resolución del sistema lineal con CR, comprobar si convergió, calcular el nuevo autovalor y por último normalizar el vector. El pseudocódigo de esto se resume en el Algoritmo 15.

Algo importante que debe considerarse es que en el algoritmo CR el hilo i-ésimo utiliza información del lugar $2i$ en la primer iteración, pero en la segunda utiliza el los elementos de la matriz en el lugar $4i$, lugar sobre el cual debería haber operado el hilo $2i$ en la primer iteración. El problema inevitable de esta implementación surge cundo el hilo $i-$ésimo, llega a la segunda iteración de CR antes que el hilo $2i-$ésimo termine la primer iteración. Sucede que el hilo $i-$ésimo estará accediendo a información falsa, provocando la falla del algoritmo. Para evitar este problema se puede utilizar funciones de sincronización disponibles en el lenguaje CUDA. Sin embargo las sincronizaciones

Algoritmo 15 Determinación de la convergencia del método de potencia inversa

1: i =Identificador del hilo
2: $q_i = 1$ ▷ Defino el vector inicial
3: **mientras** TRUE **hacer**
4: Aplicar sobre **aux** el Algoritmo 12
5: Aplicar Algoritmo 14
6: Aplicar sobre **aux** el Algoritmo 13
7: **si** $aux_0 = 0$ **entonces** ▷ Si convergió, termina el algoritmo
8: Salir del kernel
9: **fin si**
10: $aux_i = \|q_i\|^2$
11: Aplicar sobre el vector **aux** Algoritmo 13
12: $q_i = q_i / aux_0$
13: $aux_i = q_i \, (a_i \, q_{i-1} + b_i \, q_i + c_i \, q_{i+1})$
14: Aplicar sobre **aux** Algoritmo 13
15: $\lambda = aux_0$
16: **fin mientras**

funcionan sólo a nivel de bloque de hilos, ya que los bloques deben ser independientes para obtener escalabilidad en los programas. Por ello es que se utilizará un único bloque de hilos, utilizando tantos hilos como el hardware lo permita.

Al único bloque utilizado se asignaba una cantidad de hilos igual al mínimo entre la cantidad máxima de hilos por bloque máximo, la cantidad máxima de hilos por multiprocesador y la cantidad máximo de hilos que permite el tamaño de la memoria de registros.

Esto posee la obvia desventaja de no poder utilizar todos los multiprocesadores, pero el utilizar menos hilos que el número máximo disponible permite utilizar *streams*, permitiendo ejecutar el cálculo de varios autovectores al mismo tiempo en los multiprocesadores que estén desocupados. De esta forma se puede ejecutar el cálculo de otra base Sturmiana diferente en otro núcleo de la CPU y poder utilizar la GPU para el cálculo de esa base al mismo tiempo.

El algoritmo CR sobreescribe la matriz original, por lo que luego de una iteración del algoritmo CR hay que redefinirla, para esto se guardó en la memoria global la matriz original y se utilizan vectores auxiliares para guardar la copia que se utiliza en CR, también guardados en memoria global. Por otro lado, la matriz original se calcula en la CPU (y se almacena en memoria *page-locked*) y es copiada a lsa GPU luego. No se implementó un kernel que inicialize la matriz ya que habría que definir un kernel nuevo cada vez que se quiera cambiar de potencial, esto en la CPU no es necesario por que se utilizan punteros a funciones, caracteristica disponible sólo para procesadores gráficos que posean Compute Capability 3.5 o mayor.

Capítulo 3

Resultados

En este capítulo se presentan los resultados obtenidos. El trabajo se enfocó principalmente en la optimización de los tiempos de ejecución del cálculo de las bases Sturmianas. Se presentan comparaciones de los tiempos de ejecución de las distintas implementaciones del método QR, vistas en el Capitulo 2. Luego se presentan las bases obtenidas, calculadas sobre sistemas cuya soluciones son conocidas o se encuentran en material de referencia para la verificación del correcto funcionamiento. Al final del capítulo se muestran los resultados obtenidos para la eficiencia temporal del algoritmo implementado CUDA para obtener las funciones directamente en la GPU.

3.1. Obtención de autovalores

Se resolvió la ecuación (1.20) utilizando el esquema de diferencias finitas propuesto en el Capitulo 2, el cual consiste en la resolución de un problema de autovalores y autovectores para una matriz tridiagonal y simétrica. Para el estudio que se presenta a continuación se utilizó un sistema con $l = 0$, $U(r) = 0$, $E = -0{,}5$ u.a. y $V(r) = 1/r$. El método QR (Algoritmo 3) realiza un total de operaciones del orden $\mathcal{O}(N)$, cada vez que aplica una transformación ortogonal. Luego, se realizan tantas iteraciones de QR como sean necesarias para considerar que los elementos fuera de la diagonal sean nulos, de este modo, el elemento de la diagonal es el autovalor buscado. El criterio de convergencia para determinar si un elemento fuera de la diagonal es nulo consiste en evaluar si es suficientemente pequeño comparado con las diagonales. Para una matriz con diagonal a_i y b_i en la diagonal superior (Ver 2.14) se pide $|b_n| < \epsilon_M \sqrt{2}\,(|a_n| + |a_{n+1}|)$. Aún así, para valores a_n grandes en la diagonal, el criterio permite valores proporcionales para b_n y para evitar que pueda influir en los autovalores, se elige como criterio adicional $b_n < \epsilon_M$, éste depende del tipo de variable utilizada, en los cálculos presentados se utiliza $\epsilon_M = 2{,}2 \times 10^{-16}$, correspondiente al tipo de variable *double* (punto flotante de 64 bits).

Además se realizaron los cálculos utilizando como tipo de variable *float*, correspondiente a punto flotante de 32 bits, sin embargo por el gran número de operaciones realizadas la imprecisión del *float* se vuelve considerable y los resultados que se obtienen no son aceptables, con un error relativo promedio del 15 %.

Inicialmente se estudió el tiempo de ejecución de tres versiones del método QR. Todos los resultados presentados en esta sección son del algoritmo funcionando únicamente en la CPU. En todos los casos se utilizó el mismo procesador.

En el primer caso, se implementó como condición de convergencia, que todos los elementos de la sub-diagonal sean despreciables comparados con ϵ_M. Utilizando la primer variante vista en el Capítulo 2 (Ver Algoritmo 4), se define en la iteración del método un shift de Wilkinson según la última submatriz 2×2

$$\mathbf{L}_N = \begin{pmatrix} a_{N-1} & b_{N-1} \\ b_{N-1} & a_N \end{pmatrix}. \tag{3.1}$$

Se realizó la medición de los tiempos de ejecución en función del tamaño de la grilla utilizada (N) para el cálculo de todos los autovalores de la matriz. Para ello se utilizó la función del lenguaje C `gettimeofday,` que devuelve el tiempo transcurrido desde la *Epoch* establecida (primero de enero de 1970) con precisión de micro segundos.

Se observa en los tiempos de ejecución un claro ascenso de forma no lineal en el numero de iteraciones. Para cuantificar el orden del algoritmo se realizó un ajuste lineal del tipo $t = k\,N^{\alpha}$, obteniendo un valor $\alpha = (2{,}013 \pm 0{,}001)$.

Además se implementó un nuevo programa que realiza la iteración QR utilizando el Shift de Wilkinson definido anteriormente, hasta que el último elemento de la subdiagonal se anula (Ver Algoritmo 5). En esta instancia el último elemento de la diagonal es un autovalor y para encontrar el resto de autovalores alcanza con buscar los de la matriz formada de eliminar la última fila y la última columna. Para este caso se midió el tiempo de ejecución para distintos tamaños de grilla. Sobre los datos obtenidos se realizó un ajuste de la forma $t = kN^{\alpha}$ del cual se obtiene $\alpha = (2{,}013 \pm 0{,}001)$. Es interesante notar la mejora en el tiempo de ejecución, si bien en ambos casos crece de la forma $kN^{2{,}013}$, el segundo algoritmo es aproximadamente 40 veces más rápido.

La gran diferencia entre los resultados obtenidos se debe a la elección del Shift de Wilkinson. éste asegura una rápida convergencia del último elemento b_{N-1} con el cual se definió la matriz (3.1). En este caso el último elemento cumple el criterio de convergencia luego de dos iteraciones y en la mayoría de los caso sólo una es suficiente. En el primer algoritmo el Shift de Wilkinson es siempre el mismo y el único elemento para el cual está asegurada la convergencia es para el último (b_{N-1}). Para la convergencia

de b_{N-2} se observa que hacen falta alrededor de 50 iteraciones de QR, esto hace la ineficiencia del primer caso estudiado tan grande. Otro aspecto menos influyente es que al disminuir el tamaño la cantidad de operaciones de la iteración QR es menor.

Por otra parte se implementó el algoritmo recursivo, consistente en buscar el Shift de Wilkinson utilizando la submatriz 2×2 definida por:

$$\mathbf{L}_N = \begin{pmatrix} a_{N/2-1} & b_{N/2-1} \\ b_{N/2-1} & a_N/2 \end{pmatrix}. \tag{3.2}$$

Realizando la iteración QR eventualmente se anula el elemento de la subdiagonal, $b_{N/2-1}$, de este modo, la matriz queda reducida a 2 bloques, resultando en autovalores de uno de ellos que son independientes de los del otro, de la forma que se muestra en el Algoritmo 6. Esto posee particular interés debido a la disminución en la cantidad de cálculo que implica resolver únicamente los bloques que tendrían la mitad del tamaño.

Si bien los autovalores encontrados coinciden con los obtenidos por los métodos nombrados anteriormente, los altos tiempos de ejecución lo convierten en un método contraproducente. Incluso si se pudiese realizar la paralelización del método, los tiempos para producir la primer división son incluso mayores que los tiempos necesarios para resolver la matriz entera por cualquiera de los otros dos métodos. Tampoco se puede asegurar la convergencia del elemento $b_{N/2-1}$.

Para una representación más clara de las diferencias en los tiempos de ejecución, para energías negativas, de cada una de las tres variantes se muestra en la Tabla 3.1.

Variantes del método QR			
N	Algoritmo 4	Algoritmo 5	Algoritmo 6
10	$1{,}110^{-4}$s	$8{,}010^{-6}$s	$1{,}210^{-1}$s
100	$5{,}910^{-3}$s	$3{,}910^{-4}$s	15,2s
1000	0,8s	$3{,}310^{-2}$s	–
10000	110s	2,97s	–

Tabla 3.1: Comparación de tiempos de ejecución de las tres variantes del método QR. Se observa que la segunda variante es considerablemente más eficiente que las otras dos. Para la tercera variante no se observó convergencia para tamaños mayores a 100 dentro del tiempo esperado.

Sobre el algoritmo más eficiente de los tres implementados se realizaron cambios, el más importante es buscar elementos nulos fuera de la diagonal y dividirla en dos, resolviendo ambos bloques por separado, como se detalló en el Sección 2.3.1. El procedimiento es el representado en el Algoritmo 11, en este caso quitando la llamada asincrónica del cálculo del autovector.

Se espera que el algoritmo sea considerablemente más rápido ya que al dividir la matriz se evita una enorme cantidad de iteraciones del algoritmo 3 por ser menor el

tamaño. Sin embargo, la búsqueda de un elemento fuera de la diagonal que satisfaga la condición de convergencia luego de cada iteración de QR implica una cantidad de iteraciones comparable a las evitadas al disminuir el tamaño de la matriz. En la Figura 3.1 se muestra los tiempos utilizados por ambas variantes para calcular los primeros N autovalores, siendo ligeramente más eficiente el algoritmo que divide la matriz. Este resultado se debe a la costosa búsqueda mencionada anteriormente; se puede realizar la búsqueda cada cierta cantidad de iteraciones de QR para quitarle trabajo esa parte del algoritmo, pero los resultados no mejoran de forma considerable.

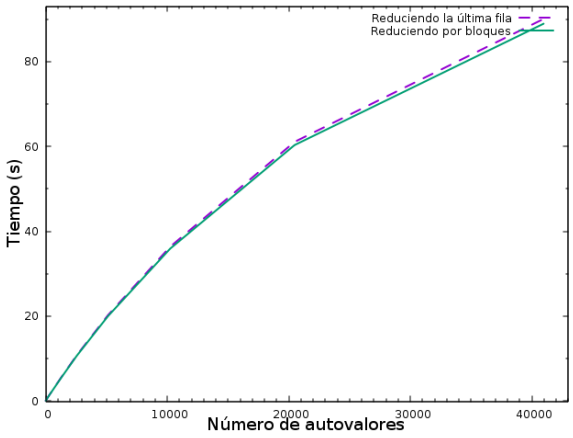

Figura 3.1: Gráfico de tiempo de ejecución en función de la cantidad de autovalores deseados, se observa una eficiencia levemente superior con la versión que divide la matriz en bloques (Ver Algoritmo 11). Todos las mediciones de tiempos fueron tomados utilizando el mismo procesador y una matriz de tamaño $N = 60000$ definiendo un sistema físico con energía $E = -2{,}0$ u.a. y masa reducida unidad, un potencial auxiliar nulo y un generador Coulombiano, utilizando condición de borde de caja.

Para cerrar el estudio de la eficiencia en el cálculo de los autovalores, se realizó una comparación entre el tiempo de ejecución del Algoritmo 11 para el cálculo de autovalores y algunas funciones disponibles en la biblioteca LAPACK[15], variando el número de autovalores requeridos. Los algoritmos utilizado por cada función son esencialmente diferentes ya que no hay ninguna rutina que permita calcular sólo algunos autovalores utilizando el método QR. Las funciones con las cuales se comparó son: `DSTEMR` y `DSTEBZ`, la primera obtiene los autovalores mediante el método MRRR[16] (o *Multiple Relatively Robust Representations* por sus siglas en inglés), mientras el segundo busca los autovalores utilizando el método de la bisección[17]. Los resultados obtenidos se muestran en la Figura 3.2, se utilizó un potencial auxiliar nulo y un generador coulombiano, energía negativa $E = -0{,}5$ u.a. y $l = 0$. No se realizaron comparaciones para energías positivas ya que no se encuentran funciones de acceso público que resuelvan el caso de una matriz tridiagonal no hermítica. Todas las funciones que posee LAPACK están implementadas para matrices tridiagonales que son, o bien reales y simétricas o bien complejas y hermíticas. En los tres casos se pidió que los autovalores fuesen obtenidos con la misma precisión y se utilizó el mismo procesador funcionando en un

único hilo.

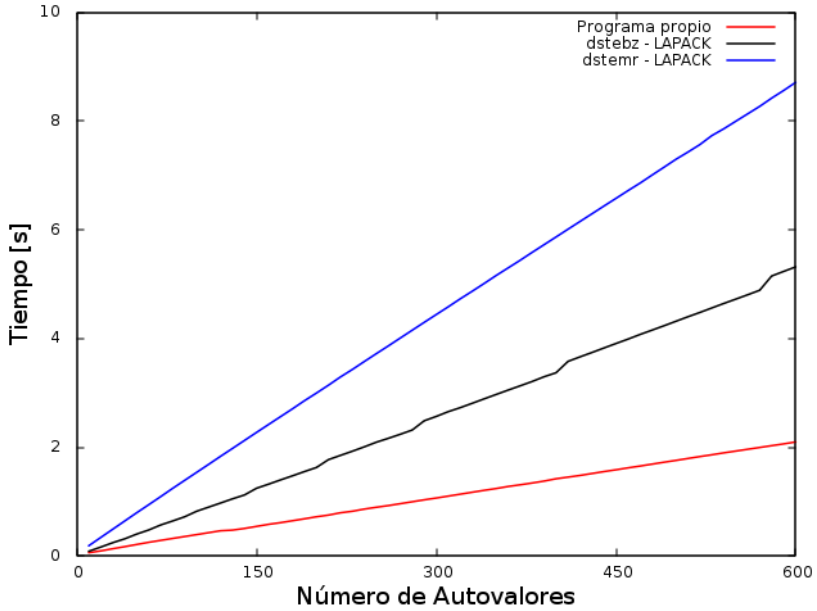

Figura 3.2: Tiempos de cálculo en función del número de elementos de la misma. Se utilizó un sistema formado por un potencial generador Coulombiano, utilizando un radio de corte $r_c = 30$ u.a., una grilla de tamaño $N = 30000$, energía $E = -0{,}5$u.a y condición de borde de caja. Los tiempos presentados corresponde a tres diferentes versiones de cálculo de autovalores del mismo sistema, utilizando la función propia y las funciones DSTEMR y DSTEBZ de la biblioteca LAPACK.

3.2. Precisión de los resultados

Se presentan a continuación los resultados del cálculo realizado para sistemas cuyas soluciones son conocidas, a modo de verificación del correcto funcionamiento de los programas implementados. Primeramente se presentarán los resultados obtenidos para sistemas con energías negativas y condiciones de borde de caja, y se muestran los valores teóricos correspondientes. Luego se muestran los resultados para energías positivas y condición de onda saliente o entrante. En todos los casos se utiliza un potencial auxiliar nulo ($U = U_{int} + Z/r$) con $U_{int} = Z = 0$, ya que son los sistemas de los cuales se conocen sus soluciones.

3.2.1. Energías negativas

Se comenzará la verificación utilizando energías negativas, en todos los casos se utilizaron condiciones de caja, esto es, la función Sturmiana vale cero más allá del radio de corte r_c.

Potencial Coulombiano

Se utilizó un potencial generador Coulombiano de la forma: $V(r) = -Z/r$, siendo $Z = 1$ u.a. la carga del potencial. Por compatibilidad con los valores de referencia[4] se utilizó un radio de corte $r_c = 30$ u.a. con una grilla de tamaño $N = 30000$, los autovalores (β_n) de este sistema son conocidos y están dados por la relación:

$$\beta_{n,l} = -(n + l)\sqrt{-2\mu\, E}, \tag{3.3}$$

En particular, utilizando $l = 0$ y $E = -0{,}5$ u.a. se obtiene que los autovalores son directamente $\beta_n = -n$.

Utilizando el algoritmo más eficiente entre los estudiados en la Sección anterior (Algoritmo 11) se calcularon los autovalores para el sistema planteado con $E = -0{,}5$ u.a. y se obtuvieron valores correctos de los mismos, que se comparan con los analíticos en la Tabla 3.2.

$\beta_n + \mathcal{O}(dr^2)$	β_n^{exacto}
-1.0000001249	-1.0
-2.0000002500	-2.0
-3.0000003752	-3.0
-4.0000005000	-4.0
-5.0000006251	-5.0
-6.0000007499	-6.0
-7.0000008779	-7.0
-8.0000010977	-8.0
-9.0000034173	-9.0
-10.0000389703	-10.0
-11.0004369417	-11.0
-12.0034928272	-12.0

Tabla 3.2: Autovalores obtenidos para el potencial generador Coulombiano con energía negativa.

La precisión obtenida en el cálculo de los autovalores depende directamente del tamaño de la discretización espacial, en especial del radio de corte (r_c). En la Tabla 3.2 se observa que la diferencia respecto al autovalor teórico aumenta cuando crece n, esto se debe principalmente al radio de corte utilizado. Por la condición de borde de caja, los autovalores tienen que ser nulos en $r = r_c$, por ello los resultados del sistema utilizado difieren del teórico, el cual asume que los autovalores decaen exponencialmente. Por otro lado se encuentra el error en el cálculo de la derivada segunda utilizando diferencias finitas, esto implica que el cálculo de los autovalores difiere del correcto en un orden dr^2.

La desventaja de aumentar el radio de corte, es que los elementos de la matriz crecen inversamente con el potencial auxiliar, es decir, aumenta el tamaño de los elementos

de la matriz. Esto resulta en un aumento importante en los tiempos de ejecución del cálculo de autovalores incluso manteniendo constante la dimensión de la matriz.

Para evidenciar la influencia del radio de corte se muestra en la Tabla 3.3 la discrepancia en el cálculo de los autovalores ($\Delta\beta_n = \beta_n - \beta_n^{exacto}$) al variar el radio de corte manteniendo constante la discretización espacial, en todos los casos $dr = 0{,}001$, y el tamaño de la grilla se varió según fuese necesario. A partir de $r_c = 30$ no se observa, a fines prácticos, una mejora en los autovalores obtenidos incluso utilizando un radio de corte $r_c = 80$. Esto indica que el error del cálculo se produce exclusivamente por la discretización espacial, de orden dr^2, además se observa que crece proporcional al número de autovalor considerado(n), esto implica un error dado por $\frac{\Delta\beta_n}{n} = 1{,}25\,10^{-7}$.

$\Delta\beta_n$u.a.					
n	$r_c = 10$u.a.	$r_c = 20$u.a.	$r_c = 30$u.a.	$r_c = 40$u.a.	$r_c = 80$u.a.
1	0,0000008604	0,0000001249	0,0000001249	0,0000001250	0,0000001250
2	0,0001031029	0,0000002499	0,0000002500	0,0000002501	0,0000002500
3	0,0035107111	0,0000003758	0,0000003750	0,0000003750	0,0000003750
4	0,0409537074	0,0000005777	0,0000005000	0,0000005001	0,0000005000
5	0,2056017831	0,0000041819	0,0000006250	0,0000006251	0,0000006250
6	0,5906278304	0,0000919631	0,0000007499	0,0000007502	0,0000007500
7	1,2282307738	0,0013608229	0,0000008778	0,0000008751	0,0000008750
8	2,1207036049	0,0118274352	0,0000010976	0,0000010002	0,0000010000

Tabla 3.3: Error del autovalor obtenido para un potencial generador Coulombiano con energía negativa $E = -0{,}5$ u.a. en función del radio de corte, manteniendo constante $dr = 0{,}001$

Por otro lado para corroborar el correcto funcionamiento del cálculo de los autoestados se muestra en la Figura 3.3 los autovectores encontrados mediante el método de la potencia inversa y la solución teórica del sistema. Se puede obtener la solución teórica del sistema considerando la ecuación de schrödinger radial (1.20) utilizando un potencial generador nulo y un potencial auxiliar coulombiano. Por simplicidad se utiliza el caso $l = 0$, resultando en la ecuación:

$$\left[-\frac{d^2}{dr^2} + \frac{\beta_n}{r} - E\right] S_{n,0}(r) = 0. \tag{3.4}$$

Dicho sistema posee dos soluciones conocidas descritas en mediante la función *hipergeométrica de Tacomi* que pueden escribirse como:

$$S^{\pm}_{n,0}(r) = \mp 2\,k\,r\,e^{\pm i\,k\,r}\,e^{\frac{\pi\,\beta_n\,\mu}{2\,k}}\,U(1 \pm i\,\frac{\beta_n\,\mu}{k}, 2, \mp 2\,i\,k\,r). \tag{3.5}$$

donde $k = \sqrt{2\,\mu\,E}$. La función hipergeométrica de Tacomi se evalúa mediante la función `gsl_sf_hyperg_U` disponible en la biblioteca científica de GNU (GLS, por sus siglas en inglés, *GNU Scientific Library*). En el caso de energía negativa, la solución $S^{-}_{n,0}(r)$ diverge para distancias grandes, por ello se descarta y se considera únicamente la

solución $S^{+}_{n,0}(r)$. Para distancias grandes la solución está dominada por un decaimiento exponencial mientras que a radios pequeños $U(a,b,r)$ es finito y por lo tanto la solución se anula por su dependencia lineal con r.

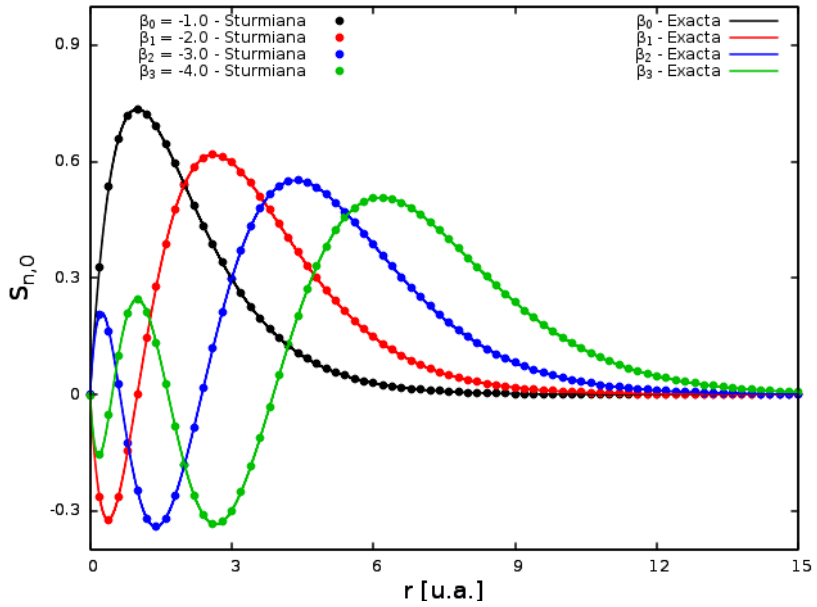

Figura 3.3: En puntos se presentan las primeras cuatro funciones Sturmianas para un potencial generador Coulombiano, utilizando un radio de corte $r_c = 30$ u.a., una grilla de tamaño $N = 30000$, energía $E = -0{,}5$ u.a y condición de borde de caja. Mientras en linea continua se muestra el resultado exacto correspondiente a un sistema con potencial Coulombiano con carga β_n.

Por otro lado, se utilizó una energía $E = -2{,}0$ u.a. para verificar que se obtienen los mismos resultados que en [8]. Se consideró el caso $l = 0$, una grilla de tamaño $N = 15000$ y un radio de corte $r_c = 15$ u.a., los autovalores exactos se obtienen utilizando la formula (3.3). Los primeros nueve autovalores obtenidos se muestran en la Tabla 3.4. A diferencia de los autovalores correspondiente a la energía estudiada anteriormente el error más influyente en los autovalores de menor magnitud es el correspondiente a la aproximación de la derivada. Esto se explica pensando en la diferencia fundamental respecto al caso anterior, esto es, la energía que en este caso es cuatro veces mayor y por lo tanto k es el doble. Esto causa que el factor exponencial en la Ecuación 3.5 sea considerablemente más pequeño y por lo tanto la condición de borde de caja se aproxima mejor a la condición real de decaimiento exponencial.

Es interesante notar que el error asociado a la discretización espacial en este caso es ocho veces el error asociado en el caso de $E = -0{,}5$, esto puede corroborarse fácilmente considerando la forma funcional de la solución exacta. Dada la solución exacta $f(r) = S^{+}_{n,0}(r)$ de la Ecuación 3.5, el error (ϵ) en el cálculo numérico de la derivada segunda está dado por:

$$\epsilon = -\frac{f'''(r)}{6}\, dr^2. \tag{3.6}$$

$\beta_n + \mathcal{O}(dr^2)$	β_n^{exacto}
-2.000001000004	-2.0
-4.000002000010	-4.0
-6.000003000054	-6.0
-8.000004000034	-8.0
-10.000005000103	-10.0
-12.000006000192	-12.0
-14.000007005784	-14.0
-16.000008194917	-16.0
-18.000013578960	-18.0

Tabla 3.4: Autovalores obtenidos para el potencial generador Coulombiano con energía negativa $E = -2$ u.a., para $N = 15000$ y $r_c = 15$ u.a.

Realizando el cambio de variable $u = k\,r$, la derivada tercera de $f(r)$ es proporcional a $-k^3$, ya que se puede cambiar de variable y tener una función $f(u)$ con $u = k\,r$, sin cambiar su forma. Considerando el sistema de energía $E = -2$ u.a. se tiene que k es dos veces mayor al correspondiente al sistema con energía $E = -0{,}5$ u.a., de este modo la constante que acompaña al dr^2 es:

$$\begin{aligned} \epsilon &= k^3\, f(u) \qquad \text{si } E = -0{,}5\,\text{u.a.} \\ \epsilon &= (2\,k)^3\, f(u) \qquad \text{si } E = -2\,\text{u.a.} \end{aligned} \tag{3.7}$$

Se puede demostrar que en ambos casos la función $f(u)$ vale igual para el mismo valor de u, esto implica que la constante que acompaña a dr^2 vale ocho veces mas que en el caso de menor energía, esto puede corroborarse con los resultados presentados en la Tabla 3.4. Por otro lado se muestran los autoestados obtenidos comparados con los estados exactos del potencial coulombiano con la nueva energía $E = -2$ u.a. obtenidos según la expresión 3.5.

Potencial de Hultén

El mismo tratamiento utilizado para estudiar el potencial Coulombiano se aplicó a un potencial generador tipo Hultén, tomando:

$$V(r) = \frac{e^{-r/a}}{1 - e^{-r/a}}, \tag{3.8}$$

donde se utilizó $a = 1$, una grilla de tamaño $N = 15000$, radio de corte $r_c = 15$ u.a. y una energía negativa de valor $E = -0{,}5$ u.a.

Los autovalores del potencial de Hultén para el caso particular $l = 0$ están determinados por[8]:

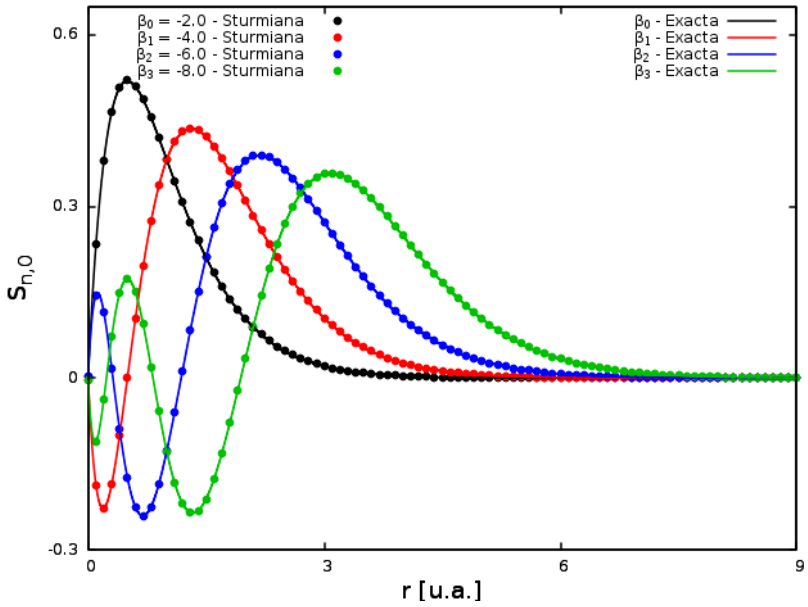

Figura 3.4: En linea de puntos se muestran primeras cuatro funciones Sturmianas para un potencial generador Coulombiano, utilizando un radio de corte $r_c = 15$ u.a., una grilla de tamaño $N = 15000$, energía $E = -2$ u.a. y condición de borde de caja. Por otro lado en linea continua se muestra el resultado exacto obtenido de manera teórica.

$$\beta_n = -\frac{-n\,(n - 2\,i\,a\,k)}{2\,a^2\,\mu}. \tag{3.9}$$

En este caso, $k = \sqrt{2\,\mu\,E} = i$ y $a = 1$, entonces los autovalores quedan determinados por $\beta_n = -\frac{n\,(n+2)}{2}$. Utilizando los parámetros mencionados anteriormente se calcularon numéricamente las soluciones obteniendo los autovalores de la Tabla 3.5.

$\beta_n + \mathcal{O}(dr^2)$	β_n^{exacto}
-1.500000583	-1.5
-4.000003249	-4.0
-7.500010499	-7.5
-12.000025833	-12.0
-17.500053750	-17.5
-24.000099750	-24.0
-31.500170334	-31.5
-40.000273001	-40.0
-49.500416252	-49.5

Tabla 3.5: Comparación entre los primeros nueve autovalores exactos y los obtenidos numéricamente para el potencial generador de Hultén con energía negativa $E = -0{,}5$ u.a. utilizando $N = 15000$ y radio de corte $r_c = 15$ u.a.

En este caso el error en el cálculo de los autovalores es debido puramente a la aproximación de la derivada segunda, aumentando el radio de corte hasta $r_c = 300$ u.a. no se observó, a fines prácticos, una mejora en los primeros autovalores. En este

caso la relación entre el autovalor y el error que acompaña al dr^2 es cuadrática, esto es, $\Delta\beta_n = a\,\beta_n^2 + b\,\beta_n + c$, se realizó un ajuste utilizando los primeros cuarenta autovalores y se obtuvo $a = (4{,}16638 \pm 0{,}00001)10^{-8}$u.a., $b = (-4{,}14 \pm 0{,}01)10^{-8}$u.a. y $c = (-9{,}1 \pm 1{,}4)10^{-08}$u.a.. La forma funcional elegida se corresponde con lo obtenido si se calcula la derivada tercera de la solución teórica ($S^+_{n,0}(r)$) para evaluar el error de la aproximación de la derivada segunda, dado por la ecuación 3.6. Esto se observa utilizando otros métodos presentes en la biblioteca LAPACK[15], tales como las rutinas `DSTEMR` y `DSTEBZ`, por lo tanto el error no proviene del algoritmo utilizado. Además se corroboró que el error es proporcional a dr^2 utilizando diferentes tamaños de matriz para un mismo radio de corte.

Por otro lado, se muestra en la Figura 3.5 los autovectores obtenidos comparados con los autoestados exactos calculados de manera teórica. Para obtener los autoestados asociados al potencial de Hultén se realiza un cambio de variable en la ecuación de Schrödinger radial (Ecuación 1.20), utilizando $U(r) = 0$ y $l = 0$ (se utiliza el caso $l = 0$ ya que es el único que posee solución analítica). De este modo la ecuación resultante es:

$$\left[-\frac{1}{2\mu}\,\frac{x^2}{a^2}\frac{d^2}{dx^2} - \frac{1}{2\mu}\,\frac{x}{a^2}\frac{d}{dx} + \frac{\beta_n\,x}{1-x} - E\right] S_{n,0}(x) = 0, \tag{3.10}$$

donde se realizó el cambio de variables $x = e^{-r/a}$. La Ecuación 3.10 posee dos soluciones linealmente independientes en términos de la *función hipergeométrica de gauss* ($_2F_1(A, B, C, z)$), estas son[18, Pág 421]:

$$S^{\pm}_{n,0}(r) = e^{\pm ikr}\,{}_2F_1(ia(-g \mp k), ia(g \mp k), 1 \mp 2ika, e^{-r/a}), \tag{3.11}$$

donde se definen los parámetros $k = \sqrt{2\mu E}$ y $g = \sqrt{k^2 + 2\mu\beta_n}$.

En el caso de estudio actual la energía del sistema es negativa y por lo tanto $k = i$. Debido a la condición de regularidad de la solución se descarta la solución $S^-_{n,0}(r)$ ya que esta diverge para radios suficientemente grandes. Por otro lado, la solución $S^+_{n,0}(r)$ decae exponencialmente con el radio por ser la función $_2F_1(A, B, C, z) = 1$ para z pequeño (y por lo tanto, para r grandes).

3.2.2. Energías positiva

Para el caso de energía positiva se estudió el potencial de Hultén, utilizando $a = 1{,}5$. El sistema se consideró con energía $E = 0{,}6$ u.a., con potencial auxiliar nulo y la condición de borde de onda saliente dada por la Ecuación (1.36). Reemplazando los parámetros de dicho sistema en la expresión 3.9, los autovalores quedan determinados por:

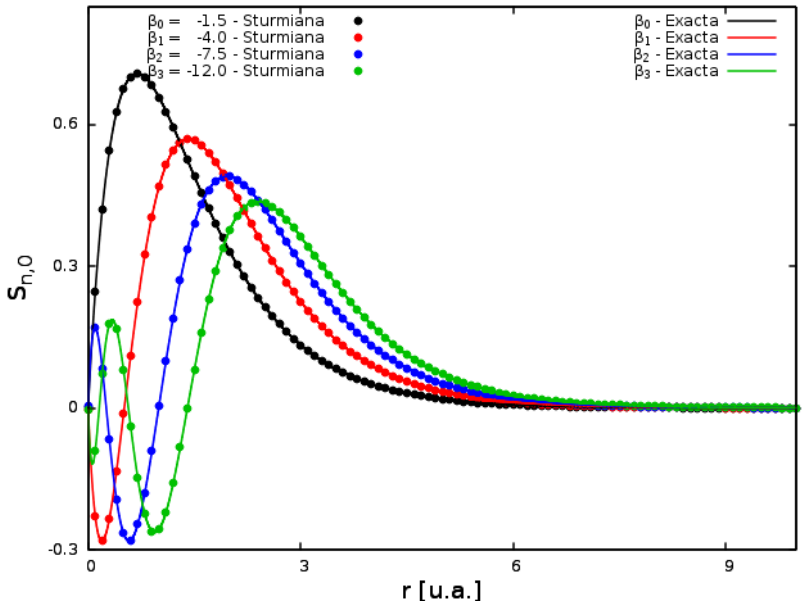

Figura 3.5: Primeras cuatro funciones Sturmianas para un potencial de Hultén graficadas en linea punteada, utilizando un radio de corte $r_c = 15$ u.a., una grilla de tamaño $N = 15000$, energía $E = -0{,}5$u.a y condición de borde de caja. Mientras en linea continua se presentan los autoestados exactos asociados a al mismo potencial con constante multiplicativa β_n.

$$\beta_n = -\frac{n^2}{4{,}5} + i\,\frac{n\sqrt{1{,}2}}{1{,}5}, \tag{3.12}$$

En la Tabla 3.6 se encuentran lo resultados numéricos obtenidos para los primeros diez autovalores comparados con los autovalores exactos.

$\beta_n + \mathcal{O}(dr^2)$	β_n^{exacto}
-0.2222218 + i 0.7302965	-0.2222222 + i 0.7302967
-0.8888904 + i 1.4605925	-0.8888888 + i 1.4605935
-2.0000014 + i 2.1908982	-2.0000000 + i 2.1908902
-3.5555406 + i 2.9211961	-3.5555555 + i 2.9211869
-5.5555324 + i 3.6514664	-5.5555555 + i 3.6514837
-8.0000227 + i 4.3817232	-8.0000000 + i 4.3817805
-10.8890220 + i 5.1120580	-10.8888888 + i 5.1120772
-14.2223997 + i 5.8425154	-14.2222222 + i 5.8423739
-18.0000720 + i 6.5729681	-18.0000000 + i 6.5726707
-22.2221012 + i 7.3033055	-22.2222222 + i 7.3029674

Tabla 3.6: Comparación entre los primeros diez autovalores obtenidos para el potencial generador de Hultén con energía positiva $E = 0{,}6$ u.a., $N = 30000$, $r_c = 30$ u.a. y condición de borde de onda saliente

Al igual que el caso del potencial de Hultén con energía negativa, no se observa una mejora en los autovalores incrementando el radio de corte, no se observaron cambios significativos incluso con radio de corte $r_c = 200$u.a.. Sin embargo se observa una

mejora al disminuir dr, lo cual indica que el error involucrado proviene únicamente de la discretización espacial.

Por otro lado, el comportamiento de los autoestados exactos descrito según la Ecuación 3.11 posee ahora ambas soluciones posibles, ya que la exponencial es imaginaria. Por ello la solución completa es una combinación lineal de $S^{+}_{n,0}(r)$ y $S^{-}_{n,0}(r)$ con factores multiplicativos $C^{\pm}$, resultando en una solución de la forma:

$$S_{n,0}(r) = C^{-}\, S^{+}_{n,0}(r) - C^{+}\, S^{-}_{n,0}(r), \tag{3.13}$$

donde las constantes $C^{\pm}$ se eligen de modo que la solución total satisfaga la condición de borde en el origen ($S_{n,0}(0) = 0$), para logra esto se utiliza $C^{\pm} = \lim_{r\to 0} S^{\pm}_{n,0}(r)$. Este límite puede ser evaluado mediante el *Teorema Hipergeométrico de Gauss*, esto es:

$$\lim_{x\to 1} {}_2F_1(a,b,c,x) = \frac{\Gamma(c)\,\Gamma(c-a-b)}{\Gamma(c-a)\,\Gamma(c-b)}, \tag{3.14}$$

donde $\Gamma(z)$ es la función Gamma, dicho teorema se cumple cuando $Re(c-a-b) > 0$ y en el caso de interés $c-a-b = 1$. De esto último se obtienen las constantes $C^{\pm}$ como:

$$C^{\pm} = \frac{\Gamma(1 \mp 2\,i\,a\,k)}{\Gamma(1 + i\,a\,g \mp i\,a\,k)\,\Gamma(1 - i\,a\,g \mp i\,a\,k)}. \tag{3.15}$$

En el sistema considerado, dichas constantes poseen la particularidad de que C^{+} es doce ordenes de magnitud menor que C^{-}. Así, la solución al sistema viene dada por $S^{+}_{n,0}(r)$ y teniendo en cuenta que $\lim_{r->\infty} {}_2F_1(a,b,c,e^{-r/a}) = 1$ se deduce que las soluciones oscilan debido a la exponencial $e^{i\,k\,r}$, independientemente del autovalor asociado.

Los resultados obtenidos para los elementos de la base Sturmiana del sistema con potencial generador tipo Hultén y potencial auxiliar nulo se muestran en la Figura 3.6 junto a los resultados exactos calculados utilizando la Ecuación 3.13. Tanto las soluciones exactas como los autoestados encontrados mediante el programa propio fueron normalizadas por su valor en $r = r_c$, esto se hace para que $S_{n,0}(r_c) = 1$ para cada n y se visualice mejor el límite asintótico en el cual todas las funciones deben tener el mismo vector de onda k.

3.3. Rendimiento

En la sección anterior se corroboró el correcto funcionamiento de los algoritmos implementados resolviendo sistemas físicos cuyas soluciones son conocidas y ya fueron estudiados en diversos trabajos. En esta nueva sección se presenta un estudio del rendimiento de los algoritmos implementados, presentando comparaciones entre el algoritmo desarrollado puramente en la CPU, y el Algoritmo 11 implementado para funcionar en

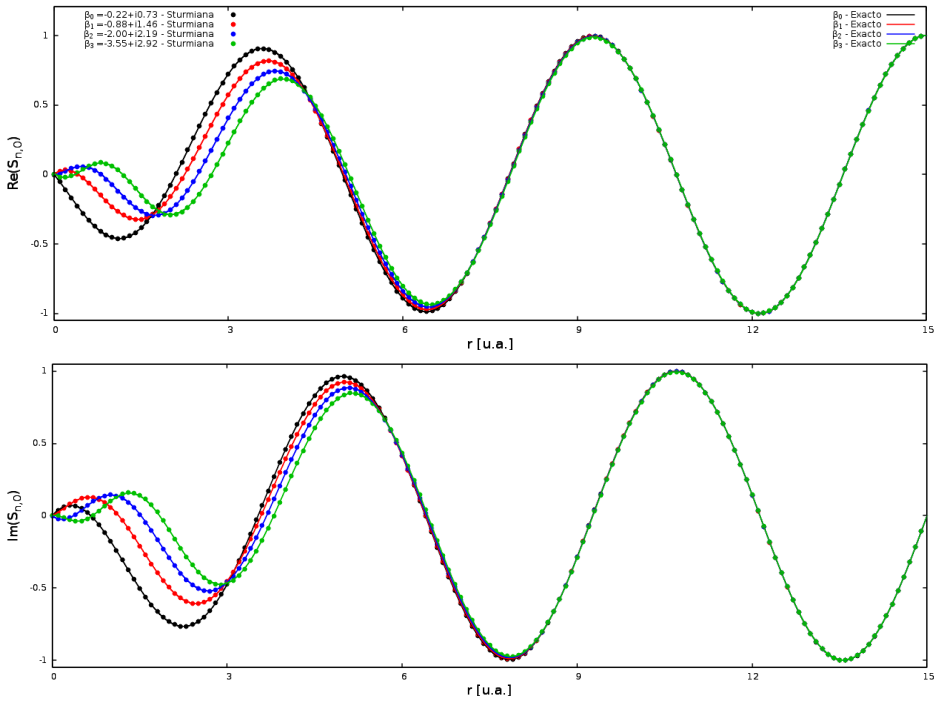

Figura 3.6: Primeras cuatro funciones Sturmianas para un potencial de Hultén, utilizando un radio de corte $r_c = 30$ u.a., una grilla de tamaño $N = 30000$, energía $E = 0{,}6$u.a y condición de borde de onda saliente. En la figura de arriba se muestra la parte real, mientras abajo se presentan la parte imaginaria de la solución.

arquitecturas heterogéneas formadas por una CPU y una GPU.

Como se explicó anteriormente la gran cantidad de solapamiento entre los datos manejados por cada hilo de la GPU hace estrictamente necesaria la sincronización de los hilos en cada iteración de la reducción cíclica. Por ello no es posible utilizar más de un bloque de hilos ya que no está asegurada la ejecución sincrónica de los bloques.

El número de hilos utilizados está impuesto por las característica de cada GPU. Por ello se elige al número de hilos que maximiza el trabajo de la GPU, pero siempre respetando las posibilidades de la misma. Hay varios factores que limitan la cantidad de hilos, entre ellos está el máximo número de hilos por multiprocesador y el máximo número de hilos por bloque. Menos directo, pero igual de importante es el número de registros por bloque, si la cantidad de hilos es demasiado grande para la memoria de registros el kernel no funcionará. Por todo esto es que se elige como número de hilos al que cumpla las tres condiciones, se toma entonces el mínimo entre las tres limitaciones nombradas anteriormente.

Para evitar tiempo en los cuales la GPU está esperando que se termine de calcular un autovector para comenzar el cálculo de otro distinto, se utilizaron *streams*, una característica de la GPU que permite enviar kernels a ejecutar asincrónicamente.

Para analizar el rendimiento se mostrarán los tiempos de ejecución de cada uno de los algoritmos implementados, tanto en CPU como en el híbrido CPU-GPU (Algoritmo 11). Además se mostrará la eficiencia temporal para distintos números de hilos por

bloque, para distintos procesadores gráficos. En todos los casos se utilizó el tipo de variable *float* para los autovectores en la GPU, mientras la CPU utiliza *double*.

Para la comparación se utilizó un potencial de Hultén con energía $E = -0{,}5$ en un grilla de $N = 30000$ elementos, utilizando condición de borde de caja en $r_c = 30$. Esto se realizó únicamente para energías negativas ya que el algoritmo que calcula los autovectores en la GPU fue implementado únicamente para matrices reales.

En la Figura 3.7 se muestran los tiempos de ejecución de los algoritmos en función del número de elementos de la base obtenida. Se midió el tiempo utilizado por la CPU para calcular sólo los autovalores (dibujada con puntos) y el tiempo del híbrido GPU-CPU para calcular los autovalores y autovectores, obteniéndose entre ellos una diferencia de tiempo del orden de 10^{-2}s, demorando por ejemplo 6,479s en la GPU contra los 6,461s utilizados para calcular sólo doscientos autovalores. Implicando esto un enorme aumento en la eficiencia comparando con el cálculo en CPU, cuyo consumo temporal para obtener cien Sturmianas es 18,737s. Aumentando linealmente esta diferencia con el número de Sturmianas deseadas. Los datos mostrados en la Figura 3.7 son obtenidos utilizando un procesador *AMD Phenom(tm) II X4 955 Processor*, de 3210MHz, mientras la GPU utilizada fue una *Tesla K40c*.

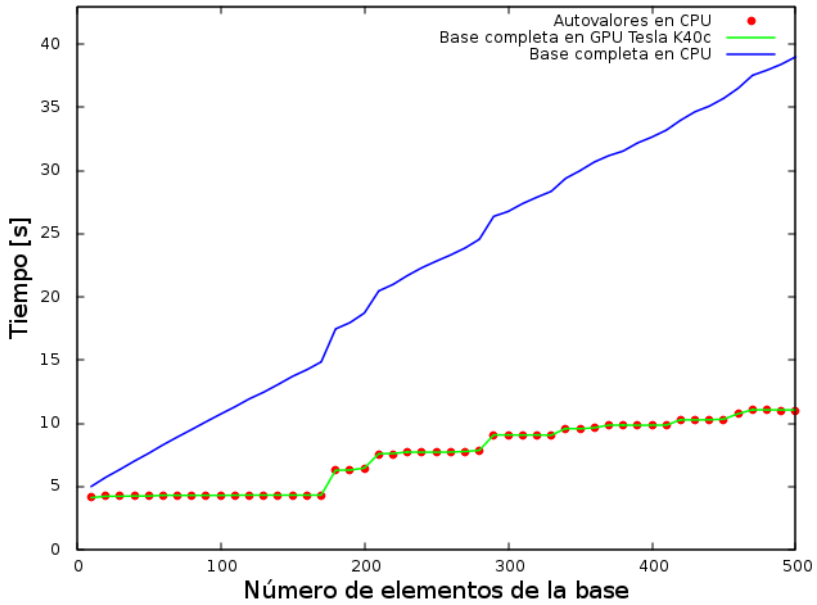

Figura 3.7: Comparación del tiempo de ejecución entre el código implementado en CPU y el implementado en CPU-GPU siendo calculado en una placa Tesla K40c. Se muestra en puntos el tiempo dedicado por la CPU al cálculo de autovalores. Se observa una gran mejora en el rendimiento utilizando la GPU.

El mismo estudio se realizo sobre una placa de vídeo *GeForce GTX 780*, cuyo rendimiento es inferior a la *Tesla K40c* y se obtuvo el mismo resultado, el cálculo de autovectores no agrega un tiempo considerable al cálculo de autovalores. En este caso el procesador utilizado fue un *Intel(R) Core(TM) i7-3820* de 3600MHz. Esto implica

que el algoritmo es incluso más eficiente que lo que se puede inferir del resultado de la Tesla K40c, ya que utilizando un procesador más potente (el cálculo demora la mitad) y una placa de menor eficiencia, se obtienen los mismos resultados. Para obtener una base de 200 elementos el híbrido CPU-GPU demora 4,033s mientras que el cálculo en la CPU de los autovalores ocupa un tiempo de 4,032s, El cálculo de la base completa en la CPU ocupa aproximadamente 2,5 veces más, con un tiempo de 10,891s.

Los resultados presentados anteriormente corresponden a utilizar el máximo número de hilos disponibles, es esperable que utilizando la mitad de hilos se obtenga un algoritmo que demore el doble del tiempo. En la Figura 3.8 se muestra el resultado del tiempo necesario para calcular las primeras cien Sturmianas variando el número de hilos utilizados para la GPU Tesla K40c. Se observa que incluso disminuyendo considerablemente el número de hilos la eficiencia del programa sigue dominada por el cálculo de autovalores empeorando en unas pocas fracciones de segundo.

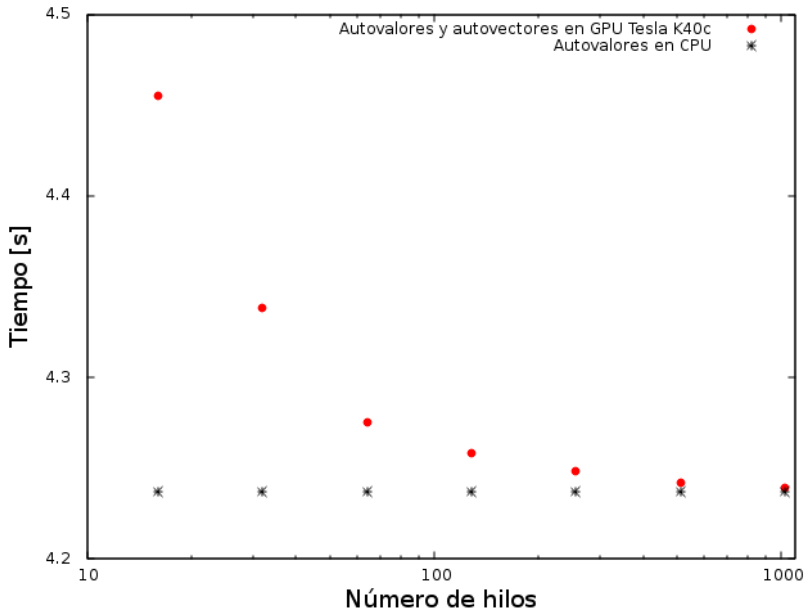

Figura 3.8: Tiempo necesario para calcular las primeras cien funciones Sturmianas variando el número de hilos en potencias de dos. Además se muestra el tiempo correspondiente al cálculo de los autovalores.

Los elementos de la base obtenidos para este sistema en el híbrido CPU-GPU coinciden con los obtenidos en la CPU y con los resultados exactos obtenidos de manera teórica. Para evitar redundancia en los resultados presentados, las figuras 3.3, 3.4 y 3.5 fueron generadas mediante el cálculo de autovectores en el híbrido CPU-GPU, donde se puede corroborar el correcto funcionamiento.

Como se vió anteriormente, el tiempo de cálculo de las bases Sturmianas se encuentra limitado por la velocidad del procesador para calcular los autovalores. El algoritmo utilizado para ello es serial y no puede ser fácilmente paralelizado para su implementación en a GPU. Sin embargo existen otros algoritmos que sí cumplen esta propiedad,

uno de ellos es el algoritmo llamado *Divide y Vencerás*. Divide la matriz y busca los autovalores por el método de la bisección, calcula todos los autovalores de matrices reales y simétricas, para matrices complejas no se asegura la convergencia. Se utilizó el programa disponible en una serie de ejemplos de programación en CUDA de NVIDIA que calcula todos los autovalores de una matriz simétrica y real mediante dicho algoritmo. No se obtuvieron resultados favorables, siendo la convergencia del algoritmo muy lenta (más de un día) debido a los enormes elementos que posee la matriz.

Capítulo 4

Conclusiones

Se desarrolló un código para el cálculo numérico de bases Sturmianas para el problema de dos cuerpos cuánticos interactuando mediante un potencial dependiente de la distancia entre ambos. Dicho potencial consiste en la suma de uno de largo alcance, llamado potencial *auxiliar* y otro de corto alcance denominado *generador*. El generador se encuentra multiplicado por un factor utilizado como autovalor de la ecuación mientras la energía se considera fija. El conjunto de autoestados asociados forman una base.

Por la simetría de los potenciales utilizados, la solución se obtiene como un producto de los Armónicos Esféricos y la solución a la ecuación radial que resulta de la separación de variables. Se discretizó el espacio y se resolvió la ecuación diferencial realizando una aproximación de orden $\mathcal{O}(dr^2)$, con dr el ancho de la discretización espacial. De este modo, la ecuación diferencial se convierte en un problema de autovalores y autovectores de una matriz tridiagonal y simétrica. Dicha matriz posee un único elemento complejo situado en la diagonal, el cual hace que la matriz no sea Hermítica.

Para la determinación de los autovalores se implementó el algoritmo QR, que consiste en aplicar sucesivas transformaciones ortogonales a la matriz original hasta obtener una forma diagonal. Para esto se implementaron distintas versiones: la primera consistió en el algoritmo QR sin modificaciones, para la segunda se mejoró disminuyendo en uno el tamaño de la matriz cada vez que se obtiene un autovalor. En la tercera versión, se buscó anular el elemento fuera de la diagonal que se encuentra en la mitad de la matriz, para dividirlo en dos bloques independientes y resolver así recursivamente ambas mitades.

La segunda versión resultó ser aproximadamente 40 veces más rápida que la primera para los potenciales utilizados. Por otro lado, la implementación recursiva fue considerablemente más lenta y no se pudo asegurar su convergencia para matrices de tamaño mayor a 100.

Sobre la segunda versión se realizó una mejora que consiste en realizar una búsqueda

sobre los elementos fuera de la diagonal, y separar la matriz en dos bloques completamente independientes. Cada bloque, de menor tamaño que la matriz original, es resuelto de forma más eficiente. Esta variante no produjo una mejora considerable en los tiempos de ejecución, llegando a ser sólo 1 % más rápido. Esto se debe al costo computacional de la búsqueda de un elemento nulo fuera de la diagonal.

En cuanto al cálculo de los autovectores asociados, se utilizó el método de la potencia inversa, el cual consiste en resolver repetida veces un sistema de la forma $\mathbf{A}\,\mathbf{x} = \mathbf{b}$. Inicialmente se utilizó el algoritmo LU para resolver dicho sistema en la CPU, luego se optimizó el cálculo implementando el método de la potencia inversa en CUDA para su funcionamiento en procesadores gráficos (GPU). Para resolver el sistema nombrado anteriormente en la GPU, se implementó el algoritmo CR (reducción cíclica). De este modo se define un programa híbrido que realiza el cálculo de autovalores en la CPU mientras el cálculo de los autovectores se realiza asincrónicamente en la GPU.

El programa está implementado de forma tal que cada vez que el método QR obtiene un autovalor comienza asincrónicamente el cálculo del autovector asociado en la GPU, así la CPU continúa con el cálculo del siguiente autovalor. El algoritmo CR se ejecuta utilizando un sólo bloque de hilos, y para maximizar el uso de la GPU se utilizaron *streams*, para poder calcular más de un autovector simultáneamente. Los resultados del híbrido CPU-GPU fueron favorables, no se observó una diferencia mayor al 0,05 % entre el tiempo necesario para calcular la base completa y el utilizado para calcular únicamente los autovalores. Por otro lado, el algoritmo serial funcionando en CPU posee un crecimiento lineal en el tiempo de cálculo, demorando aproximadamente 2,5 veces más tiempo para la obtención de una base de 200 elementos, y creciendo esta diferencia linealmente. Además se estudió la eficiencia en el cálculo de una base de 100 elementos cambiando el número de hilos, resultando la variación del orden de 0,2s en 4,24s entre utilizar 1024 o 16 hilos.

Los resultados recién mencionados suponen una mejora importante en la eficiencia de la resolución del problema de tres cuerpos cuánticos, el cual se resuelve utilizando una combinación lineal de los elementos de la base Sturmiana del problema de dos cuerpos.

Bibliografía

[1] Sherrill, C. D. An introduction to configuration interaction theory. *School of Chemistry and Biochemistry, Georgia Institute of Technology*, 1995.

[2] Bransden, B., Joachain, C. Physics of Atoms and Molecules. Pearson Education. Prentice Hall, 2003.

[3] Gasaneo, G., Mitnik, D., Frapiccini, A., Colavecchia, F., Randazzo, J. Theory of hyperspherical sturmians for three-body reactions. *The Journal of Physical Chemistry A*, **113** (52), 14573–14582, 2009.

[4] Randazzo, J. Métodos ab-initio para el problema de tres cuerpos. Tesis Doctoral, PhD thesis, Instituto Balseiro, Universidad Nacional de Cuyo, Argentina, 2009.

[5] Ambrosio, M., Del Punta, J., Rodriguez, K., Gasaneo, G., Ancarani, L. Mathematical properties of generalized sturmian functions. *Journal of Physics A: Mathematical and Theoretical*, **45** (1), 015201, 2011.

[6] Landau, L. D., Lifshitz, E. M. Quantum Mechanics, Non-Relativistic Theory: Vol. 3 of Course of Theoretical Physics. AIP, 1958.

[7] Golub, G. H., Van Loan, C. F. Matrix computations, tomo 3. JHU Press, 2012.

[8] Mitnik, D. M., Colavecchia, F. D., Gasaneo, G., Randazzo, J. M. Computational methods for generalized sturmians basis. *Computer Physics Communications*, **182** (5), 1145–1155, 2011.

[9] Bindel, D., Demmel, J., Kahan, W., Marques, O. On computing givens rotations reliably and efficiently. *ACM Transactions on Mathematical Software (TOMS)*, **28** (2), 206–238, 2002.

[10] Wilkinson, J. H. Global Convergence if Tridiagonal QR Algorithm with Origin Shifts. *Linear algebra and its applications*, **1**, 409–420, 1968.

[11] Wang, T.-L. Convergence of the tridiagonal qr algorithm. *Linear algebra and its applications*, **322** (1-3), 1–17, 2001.

[12] Gander, W., Golub, G. H. Cyclic reduction—history and applications. *Scientific computing (Hong Kong, 1997)*, págs. 73–85, 1997.

[13] NVIDIA. NVIDIA CUDA Programming Guide 8.0. 2017. `https://docs.nvidia.com/cuda/cuda-c-programming-guide/`.

[14] NVIDIA. NVIDIA CUDA Toolkit Documentation - cuSPARSE. 2017. `http://docs.nvidia.com/cuda/cusparse/`.

[15] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., *et al.* LAPACK Users' Guide. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[16] Dhillon, I. S., Parlett, B. N. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra and its Applications*, **387**, 1–28, 2004.

[17] Kahan, W. Accurate eigenvalues of a symmetric tri-diagonal matrix. Inf. téc., DTIC Document, 1966.

[18] Newton, R. G. Scattering theory of waves and particles. Springer Science & Business Media, 2013.